Gradient-based Constrained Optimization Using a Database of Linear Reduced-Order Models

Youngsoo Choi, David Amsallem, and Charbel Farhat

Abstract

A novel methodology for accelerating the solution of PDE-constrained optimization is introduced. It is based on an offline construction of database of local ROMs and an online interpolation within the database. The online flexibility of the ROM database approach makes it amenable to speeding-up optimization-intensive applications such as robust optimization, multi-objectives optimization, and multi-start strategies for locating global optima. The accuracy of the ROM database model can be tuned in the offline phase where the database of local ROMs is constructed through a greedy procedure. In this work, a novel greedy algorithm based on saturation assumption is introduced to speed-up the ROM database construction procedure. The ROM database approach is applied to a realistic wing design problems and leads to a large online speed-up.

Key words. aeroelasticity; flutter; interpolation on manifolds; model reduction; gradient-based optimization

1 Introduction

Many physical and social phenomena can be described by Partial Differential Equations (PDEs) and accurately simulated thanks to advances in numerical analysis and computer technology. PDE-constrained optimization problems arise in numerous important applications: design optimization, inverse problems, and optimal control. In spite of the importance of PDE-constrained optimization, many difficulties are known for the process of solving it. First of all, it is hard to find a robust PDE solver that works for dramatic changes in parameter values. This issue, although it has been resolved to a certain extent due to active research on this topic, remains an ongoing research topic as the complexity of applications requiring PDE modeling increases. Second, a PDE solve can be very expensive for complex problems. This causes the optimization process to be impractically long due to the many queries to the PDE solver involved. This second difficulty can be resolved by replacing the PDE with a surrogate model such as a projection-based Reduced Order Model (ROM) that lies within the subspace spanned by a Reduced Order Basis (ROB) [34, 29]. Unfortunately, it has been shown that ROMs that are constructed for a given value of parameters are in general not robust with respect to parameter changes [15].

In the context of optimization, the robustness of a ROM can be addressed in three different ways. First, a global ROM that is globally accurate in the parameter space can be constructed offline and used in the optimization process online [8, 37, 26]. Because there is no call to the PDE solver in the online phase, the optimization process can be accelerated tremendously. However, the accuracy of the global ROM over the whole parameter space depends heavily on the size of the ROB. The bigger the parameter space is, the larger the size of ROB must be to have a sufficient accuracy on the parameter space. The second way of improving the robustness of a ROM in optimization is to adaptively update the reduced-order basis of the global ROM [38, 39, 40, 16]. As the optimization progresses, some basis vectors in the ROB are eliminated and some new vectors are added in order to improve the accuracy of the global ROM around the current point. Due to the locality of the global ROM, this progressive approach gives a better accuracy than using one global ROM over the whole parameter space. However, the optimization process becomes slow because updating the global ROM adaptively requires calling the PDE solves.

The third way of improving the robustness of a ROM in optimization is to use a database of local parameterized ROMs and interpolate the ROMs to quickly generate the new ROMs at non-populated data points [3, 1]. The attractiveness of this approach is two-fold: the availability of local ROMs with a small size of ROB over the whole parameter space and the avoidance for constructing a new ROM within the optimization process. Additionally, the ROM database approach gives great online flexibility. For example, the ROM database approach is appealing when the multiple optimization solves are necessary such as robust optimization, multi-objectives optimization, and global optimization with multiple starts. It is because the database of local ROMs can be reused without additional offline phase. In the context of multidisciplinary problems, different databases from different disciplinary can also be easily combined. Finally, the fast online phase in the ROM database approach enables real-time optimization for time-critical applications.

This paper develops a methodology for a ROM database approach in the optimization context. A novel, cost-efficient greedy algorithm for constructing a database is introduced and compared with existing greedy algorithms. The new greedy algorithm can be applicable not only to the construction of local ROM database, but also to constructing a global ROM where snapshots from parameter space need to be chosen in a smart way. It also explains how to obtain ROMs and their sensitivities at non populated data points. Finally, the proposed database approach is applied to a realistic wing design problems.

The rest of the paper is organized as follows. In Section 2, the optimization problem of interest and projection-based model reduction are presented. The main idea of the proposed methodology for solving a PDE-constrained optimization is described in Section 3. A new approach for constructing a database $\mathcal{DB}$ is presented and compared with existing approaches in Section 4. Section 5 presents the consistent interpolation of ROMs on matrix manifolds and derives the gradients of the ROM interpolant with respect to design parameters $\boldsymbol{\mu}$. Section 6 demonstrates the solution procedure for the ROM database model-constrained optimization in an aeroelastic wing shape optimization. Section 7 concludes the paper.

2 Problem statement

The following optimization problem is considered:

$$\boxed{\begin{array}{ll} \underset{\boldsymbol{\mu}\in\mathcal{D}}{\text{minimize}} & f(\mathbf{w}(\boldsymbol{\mu}),\boldsymbol{\mu}) \\ \text{subject to} & \mathbf{c}(\mathbf{w}(\boldsymbol{\mu}),\boldsymbol{\mu}) \leq \mathbf{0} \end{array}} \tag{1}$$

where $f(\cdot,\cdot) \in \mathbb{R}$ is an objective function, $\mathbf{c}(\cdot,\cdot) \in \mathbb{R}^{N_c}$ defines N_c inequality constraints, $\boldsymbol{\mu} \in \mathcal{D} \subset \mathbb{R}^{N_\mu}$ is a vector of N_μ optimization variables, $\mathcal{D}$ is a parameter space, and $\mathbf{w} \in \mathbb{R}^{N_w}$ is a vector of N_w state variables solution of a parameterized linear system:

$$\mathbf{A}(\boldsymbol{\mu})\mathbf{w}(\boldsymbol{\mu}) = \mathbf{b}(\boldsymbol{\mu}). \tag{2}$$

This linear system typically arises from the linearization of a non-linear PDE around a nominal condition. The optimization problem (1) only handles the parameter vector $\boldsymbol{\mu}$ and not the vector of state variables $\mathbf{w}$. This problem therefore pertains to the framework of Nested Analysis and Design (NAND). Alternative framework for solving a PDE-constrained optimization is Simultaneous Analysis and Design (SAND) [8, 11, 13]. In a gradient-based optimization algorithm, the first derivatives of the objective function $f(\mathbf{w}(\boldsymbol{\mu}),\boldsymbol{\mu})$ and each of the constraints $c_i(\mathbf{w}(\boldsymbol{\mu}),\boldsymbol{\mu})$ for $i = 1,\cdots,N_c$ need to be computed. In general, if q denotes a generic quantity of interest whose derivative with respect to $\boldsymbol{\mu}$ is required such as f and c_i, the chain rule leads to

$$\frac{dq}{d\mu_i}(\mathbf{w}(\boldsymbol{\mu}),\boldsymbol{\mu}) = \frac{\partial q}{\partial \mu_i}(\mathbf{w}(\boldsymbol{\mu}),\boldsymbol{\mu}) + \frac{\partial q}{\partial \mathbf{w}}(\mathbf{w}(\boldsymbol{\mu}),\boldsymbol{\mu})\frac{\partial \mathbf{w}}{\partial \mu_i}(\boldsymbol{\mu}). \tag{3}$$

Equation (2) are also differentiated for each parameter μ_i, $i = 1\cdots,N_\mu$, leading to a linear system:

$$\mathbf{A}(\boldsymbol{\mu})\frac{\partial \mathbf{w}}{\partial \mu_i}(\boldsymbol{\mu}) = \frac{\partial \mathbf{b}}{\partial \mu_i}(\boldsymbol{\mu}) - \frac{\partial \mathbf{A}}{\partial \mu_i}(\boldsymbol{\mu})\mathbf{w}(\boldsymbol{\mu}). \tag{4}$$

Substituting the sensitivity solution of (4) into (3) leads to

$$\frac{dq}{d\mu_i}(\mathbf{w}(\boldsymbol{\mu}),\boldsymbol{\mu}) = \frac{\partial q}{\partial \mu_i}(\mathbf{w}(\boldsymbol{\mu}),\boldsymbol{\mu}) + \left(\frac{\partial q}{\partial \mathbf{w}}(\mathbf{w}(\boldsymbol{\mu}),\boldsymbol{\mu})\right)\mathbf{A}(\boldsymbol{\mu})^{-1}\left(\frac{\partial \mathbf{b}}{\partial \mu_i}(\boldsymbol{\mu}) - \frac{\partial \mathbf{A}}{\partial \mu_i}(\boldsymbol{\mu})\mathbf{w}(\boldsymbol{\mu})\right). \quad (5)$$

There are two approaches for computing the set of sensitivities $\{\frac{dq}{d\mu_i}(\mathbf{w}(\boldsymbol{\mu}),\boldsymbol{\mu})\}_{i=1}^{N_\mu}$.

1. In the *direct approach*, the state sensitivities $\frac{\partial \mathbf{w}}{\partial \mu_i}(\boldsymbol{\mu})$ are first computed by solving the linear system (5), then the sensitivities $\frac{dq}{d\mu_i}$ can be evaluated by (3).

2. In the *adjoint approach*, the adjoint vector $\boldsymbol{\lambda}_q(\boldsymbol{\mu})$ is first computed by solving the following linear system $\mathbf{A}(\boldsymbol{\mu})^T\boldsymbol{\lambda}_q(\boldsymbol{\mu}) = \frac{\partial q}{\partial \mathbf{w}}(\mathbf{w}(\boldsymbol{\mu}),\boldsymbol{\mu})$ and all the sensitivities $\frac{dq}{d\mu_i}$ are compute as

$$\frac{dq}{d\mu_i}(\mathbf{w}(\boldsymbol{\mu}),\boldsymbol{\mu}) = \frac{\partial q}{\partial \mu_i}(\mathbf{w}(\boldsymbol{\mu}),\boldsymbol{\mu}) + \boldsymbol{\lambda}_q(\boldsymbol{\mu})^T\left(\frac{\partial \mathbf{b}}{\partial \mu_i}(\boldsymbol{\mu}) - \frac{\partial \mathbf{A}}{\partial \mu_i}(\boldsymbol{\mu})\mathbf{w}(\boldsymbol{\mu})\right),\ i = 1,\cdots,N_\mu. \quad (6)$$

The direct approach requires N_μ linear solves while the adjoint approach requires $N_c + 1$ solutions. Hence, if $N_\mu \leq 1+N_c$ the direct approach is preferable while the adjoint approach is preferable otherwise. Problem (1) can then be solved by a gradient-based nonlinear optimization algorithm such as Sequential Quadratic Programming (SQP) [19] together with a quasi-Newton approximation of the Hessian matrix, the trust-region method [14], or the interior-point method [36].

The solution of the linearized PDE and its associated sensitivity or adjoint equations is computationally expensive as it requires the solution of linear systems of dimension N_w. To alleviate that cost, projection-based model reduction reduces the dimension of the system to be solved by reducing the dimensionality of the state $\mathbf{w}(\boldsymbol{\mu})$. For that purpose, a pre-computed reduced-order basis (ROB) $\mathbf{V}(\boldsymbol{\mu}) \in \mathbb{R}^{N_w\times k}$ spanning a k-dimensional subspace $\mathcal{S}(\boldsymbol{\mu}) \subset \mathbb{R}^{N_w}$ is defined and the state approximated as:

$$\mathbf{w}(\boldsymbol{\mu}) \approx \mathbf{V}(\boldsymbol{\mu})\mathbf{w}_r(\boldsymbol{\mu}), \quad (7)$$

where $k \ll N_w$ and $\mathbf{w}_r(\boldsymbol{\mu}) \in \mathbb{R}^k$ denotes the reduced coordinates of the state $\mathbf{w}(\boldsymbol{\mu})$ in terms of the ROB $\mathbf{V}(\boldsymbol{\mu})$. The state approximation (7) introduces a (usually) non-zero residual $\mathbf{r}(\mathbf{w}_r,\boldsymbol{\mu})$ associated with the parameterized linear system (2) defined as:

$$\mathbf{r}(\mathbf{w}_r,\boldsymbol{\mu}) = \mathbf{A}(\boldsymbol{\mu})\mathbf{V}(\boldsymbol{\mu})\mathbf{w}_r - \mathbf{b}(\boldsymbol{\mu}). \quad (8)$$

This residual is then enforced to be orthogonal to a second ROB, $\mathbf{W}(\boldsymbol{\mu}) \in \mathbb{R}^{N_w\times k}$, as $\mathbf{W}(\boldsymbol{\mu})^T\mathbf{r}(\mathbf{w}_r,\boldsymbol{\mu}) = 0$ resulting in the reduced linear system of dimension k:

$$\mathbf{A}_r(\boldsymbol{\mu})\mathbf{w}_r(\boldsymbol{\mu}) = \mathbf{b}_r(\boldsymbol{\mu}), \quad (9)$$

where the parameterized reduced order matrix $\mathbf{A}_r$ and the reduced vector $\mathbf{b}_r$ are defined as $\mathbf{A}_r(\boldsymbol{\mu}) = \mathbf{W}(\boldsymbol{\mu})^T\mathbf{A}(\boldsymbol{\mu})\mathbf{V}(\boldsymbol{\mu})$ and $\mathbf{b}_r(\boldsymbol{\mu}) = \mathbf{W}(\boldsymbol{\mu})^T\mathbf{b}(\boldsymbol{\mu})$, respectively. This results in a Petrov-Galerkin projection. If $\mathbf{W}(\boldsymbol{\mu}) = \mathbf{V}(\boldsymbol{\mu})$, this is a Galerkin projection. The optimization problem (1) can then be replaced by the following cheaper problem involving the reduced coordinates only:

$$\boxed{\begin{array}{ll} \underset{\boldsymbol{\mu}\in\mathcal{D}}{\text{minimize}} & f(\mathbf{V}(\boldsymbol{\mu})\mathbf{w}_r(\boldsymbol{\mu}),\boldsymbol{\mu}) \\ \text{subject to} & \mathbf{c}(\mathbf{V}(\boldsymbol{\mu})\mathbf{w}_r(\boldsymbol{\mu}),\boldsymbol{\mu}) \leq \mathbf{0} \end{array}} \quad (10)$$

where $\mathbf{w}_r(\boldsymbol{\mu})$ is a solution of the reduced linear system of equtions, $\mathbf{A}_r(\boldsymbol{\mu})\mathbf{w}_r(\boldsymbol{\mu}) = \mathbf{b}_r(\boldsymbol{\mu})$. Constructing a set of reduced bases $(\mathbf{V}(\boldsymbol{\mu}),\mathbf{W}(\boldsymbol{\mu}))$ for a given parameter $\boldsymbol{\mu}$ can be done by Proper Orthogonal Decomposition (POD) [34], Balanced Truncation [29] or Moment Matching [18]. However, all of these approaches involve intensive computations in order to construct $(\mathbf{V}(\boldsymbol{\mu}),\mathbf{W}(\boldsymbol{\mu}))$. To address this issue, ROB and ROM interpolation approaches have been developed in [3, 1]. All of these approaches proceed by pre-computing a database of reduced operators

$$\mathcal{DB} = \{(\boldsymbol{\mu}_c,\mathbf{V}(\boldsymbol{\mu}_c),\mathbf{W}(\boldsymbol{\mu}_c))\}_{c=1}^{N_p} \quad \text{or} \quad \mathcal{DB} = \{(\boldsymbol{\mu}_c,\mathbf{A}_r(\boldsymbol{\mu}_c),\mathbf{b}_r(\boldsymbol{\mu}_c))\}_{c=1}^{N_p}. \quad (11)$$

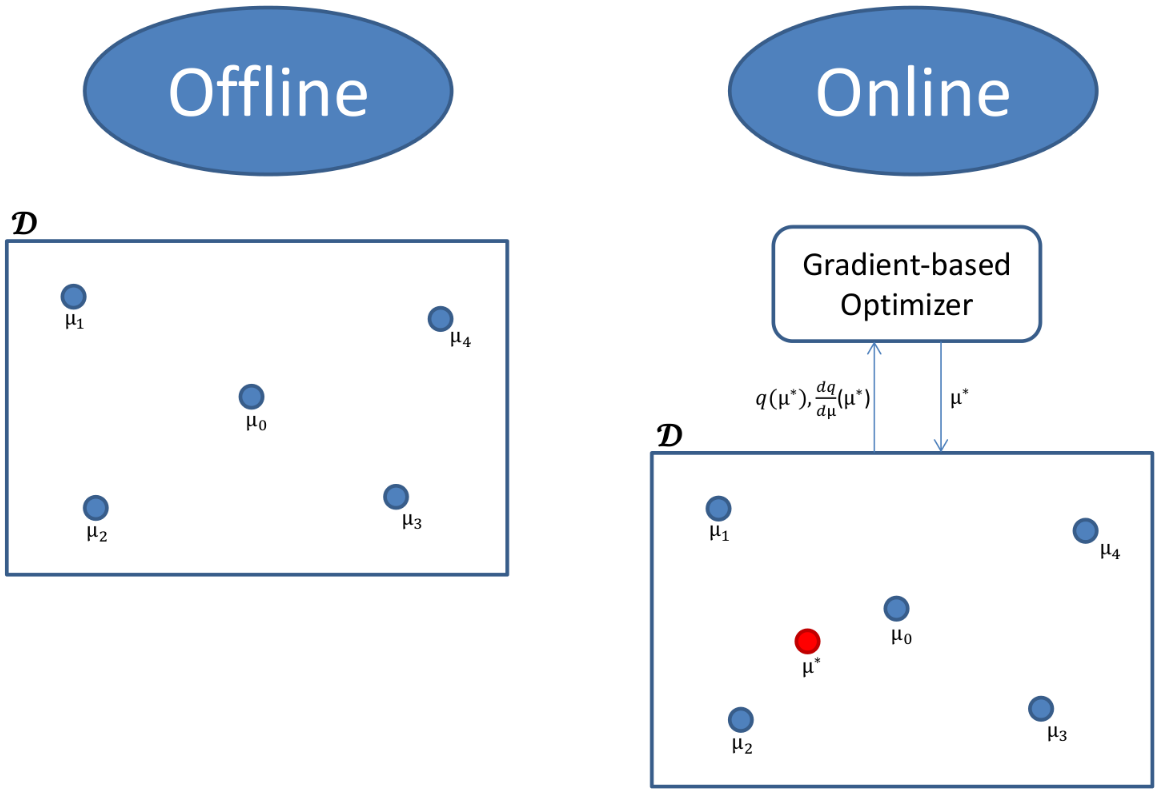


Figure 1: Methodology flow chart for the offline and online phases

The set $\mathcal{DB}$ is then interpolated to cheaply construct reduced order bases $\mathbf{V}(\boldsymbol{\mu}^\star)$ and $\mathbf{W}(\boldsymbol{\mu}^\star)$ or reduced order models $\mathbf{A}_r(\boldsymbol{\mu}^\star)$ and $\mathbf{b}_r(\boldsymbol{\mu}^\star)$ for a parameter $\boldsymbol{\mu}^\star \in \mathcal{D}$. A database of ROBs $\{(\mathbf{V}(\boldsymbol{\mu}_i), \mathbf{W}(\boldsymbol{\mu}_i))\}_{c=1}^{N_p}$ can be interpolated on the Grassmannian manifold [2]. However, the interpolation of the ROBs is more expensive than the interpolation of the ROMs because the interpolation on the Grassmannian manifold requires Singular Value Decompositions (SVDs) of matrices scaling with the size N_w. Therefore, this paper focuses on the inexpensive interpolation of the ROMs involving reduced operators only.

In order to solve PDE-constrained Optimization (1) efficiently with sufficient accuracy, the ROM-constrained optimization problem (10) can be solved, instead, using the database approach where a ROM interpolation strategy is used for robustness and efficiency. Section 3 presents the proposed methodology of solving the ROM-constrained optimization (10).

3 ROM-constrained optimization

The proposed procedure for solving the ROM-constrained optimization problem (10) can be divided into two separate phases: an offline phase followed by an online phase. In the offline phase, the ROM-database $\mathcal{DB}$ is constructed. In the online phase, the ROM-constrained optimization problem (10) is then solved by a gradient-based optimization algorithm where the function values and its derivatives are computed by ROM interpolation of the elements in the database $\mathcal{DB}$. Figure 1 schematically describes the methodology flow chart both for the offline and online phases. In the figure, the offline phase chooses six points in $\mathcal{DB}$. At each point in $\mathcal{DB}$, the corresponding local ROMs are stored (see Eq. (11)). These ROMs are then interpolated in the online phase in order to obtain ROMs at $\boldsymbol{\mu}^* \notin \mathcal{DB}$ as in Figure 1. The interpolant ROMs at $\boldsymbol{\mu}^*$ is then used to compute and pass $q(\boldsymbol{\mu}^*)$ and $\frac{dq}{d\boldsymbol{\mu}}(\boldsymbol{\mu}^*)$ to a gradient-based optimizer and the optimizer iterates until it converges.

The efficiency and accuracy of the methodology depends on how well the database is constructed in the offline phase. For example, if a large number of points of $\mathcal{D}$ are included in $\mathcal{DB}$, the accuracy of the model raises up to the level of the local ROM accuracy, but the efficiency is small as the offline phase is extremely expensive. On the other hand, if there are very few points in $\mathcal{DB}$, then both the offline and online phases are fast, but the accuracy of the ROM is low. These two extreme cases illustrate the importance of an "optimal" database construction. The "optimal" database $\mathcal{DB}^\star$ can be abstractly defined as the one that maximizes the product of efficiency and accuracy:

$$\mathcal{DB}^\star = \underset{\mathcal{DB}}{\operatorname{argmax}} \ \ \text{efficiency}(\mathcal{DB}) \cdot \text{accuracy}(\mathcal{DB}). \tag{12}$$

Finding the optimal database $\mathcal{DB}^\star$ is a challenging task in the offline phase.

4 Database construction

There are two main approaches for constructing a database: by a priori sampling approach and by adaptive sampling. The a priori sampling approach requires no information about error or accuracy of the model on $\mathcal{D}$ whereas the adaptive sampling approach requires knowledge about error of the model. The a priori sampling approach tries to select samples in $\mathcal{DB}$ so that $\mathcal{DB}$ is a good representation of $\mathcal{D}$ topologically or statistically. Examples of a priori sampling include full factorial design and latin hypercube sampling. On the other hand, the adaptive sampling approach updates $\mathcal{DB}$ in a way that the update produces the "maximum" increase in accuracy of the model. Greedy algorithms are widely used in the adaptive sampling approach [8, 9, 10, 21, 24, 31, 35]. Because the a priori sampling approach does not depend on the model, it leads to a fast construction of the database but tends to include unnecessary samples. On the other hand, the adaptive sampling approach tends to construct a database that is closer to an optimal database. However, the procedure of adaptive sampling is more computationally expensive than the one of a priori sampling approach because the adaptive sampling approach relies on repeated evaluations of the errors of the model (or some error indicators) to assess the accuracy of the model.

Section 4.1 briefly summarizes the full factorial design and latin hypercube sampling approaches. Section 4.2 describes the proposed adaptive sampling technique based on a greedy algorithm. Both sections focus on aspects of the sampling methods in the context of the ROM database model.

4.1 A priori sampling

In a full factorial design, each variable domain is divided into several factor levels. Every combination of the factor levels is then included in the database $\mathcal{DB}$. An example of full factorial design in $\mathbb{R}^2$ is depicted in Figure 2 (left) where all the factor levels are set to five. By the nature of the design, all the points in $\mathcal{DB}$ produced by full factorial design are uniformly spaced, which is a desired property in the context of interpolation because interpolation of points with irregular spacing may result in an ill-conditioned system of equations. However, the number of points in $\mathcal{DB}$ increases exponentially as the size of the parameter space N_μ increases. For example, if $\mathcal{D} \subset \mathbb{R}^6$ and each variable's factor level is three, then 729 points are in $\mathcal{DB}$ which becomes expensive. Additionally, it is hard to determine a priori the appropriate size of the database for a certain accuracy of the model because the full factorial design does not use any information about the accuracy of the model.

Latin hypercube sampling tries to overcome the issue of oversampling in $\mathcal{DB}$ by randomly generating points in a way that the generated points are well distributed in $\mathcal{DB}$. In latin hypercube sampling, each variable in $\mathcal{D} \subset \mathbb{R}^{N_\mu}$ is uniformly divided into a same number of partitions (e.g., M partitions for each variable). It then enforces to have one sample point in each dimension (see Figure 2 (right)). Note that the number of points in $\mathcal{DB}$ only depends on the number of partitions, M, not on N_μ. This is a desirable property in order to avoid the curse of dimensionality. However, the accuracy of the model now strongly depends on M. The higher the M value is, the more accurate the model is. Thus, latin hypercube sampling does not provide a complete freedom from the curse of dimensionality because the higher N_μ is, the higher M is required for the accuracy of the model. Additionally, the points in $\mathcal{DB}$ generated by latin hypercube sampling are not guaranteed to have even spacing. This may cause issues in accuracy of the ROM database approach away from sample points. Also, latin hypercube sampling is unlikely to choose corner points, so some points in $\mathcal{D}$ have to be evaluated by extrapolation only. Finally, it is

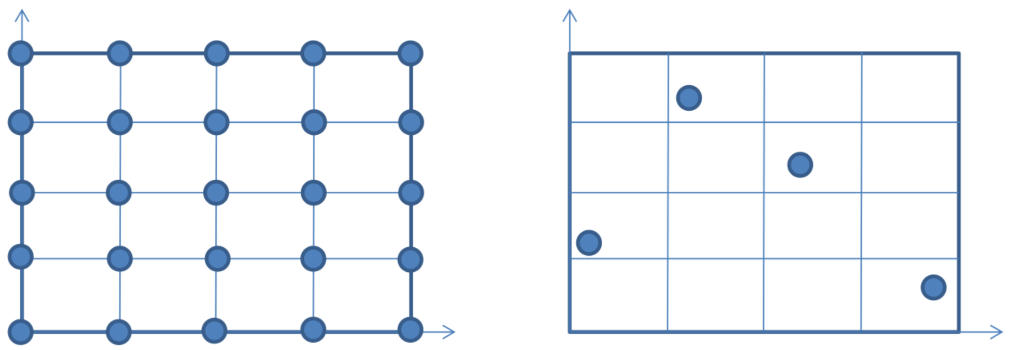


Figure 2: Two a priori sampling approaches: full factorial design (left), latin hypercube sampling (right).

hard to know a priori the appropriate M value for a certain accuracy of the model as in the case of full factorial design as latin hypercube sampling does not use any information about the accuracy of the model.

4.2 Adaptive sampling

4.2.1 Classical greedy procedure

Adaptive sampling approaches make use of error estimates associated with the ROM to populate the database $\mathcal{DB}$ iteratively. Greedy algorithms are widely used as simple adaptive sampling approach. At each iteration, the greedy algorithm selects a point where the maximum error occurs in $\mathcal{D}$ and reduces the error by including the point in $\mathcal{DB}$. More formally, denoting by $\mathcal{DB}_{N_p}$ a ROM database with N_p samples, let $\varrho(\boldsymbol{\mu}; \mathcal{DB}_{N_p})$ denote an error indicator for the $\mathcal{DB}_{N_p}$-based ROM-interpolation model at $\boldsymbol{\mu} \in \mathcal{D}$. Let Ξ also define a candidate set of N_Ξ test points in $\mathcal{D}$. The candidate set Ξ must include enough points to represent $\mathcal{D}$ well. One can either generate Ξ by full factorial design with a large value of factor level for each variable or by latin hypercube sampling with a large value of M. The greedy algorithm first picks a point $\boldsymbol{\mu}_1$ in Ξ and builds the initial database $\mathcal{DB}_1 = \{\mathbf{A}_r(\boldsymbol{\mu}_1), \mathbf{b}_r(\boldsymbol{\mu}_1)\}$. At iteration N_p, the algorithm computes $\varrho(\boldsymbol{\mu}_i; \mathcal{DB}_{N_p})$ for $i = 1, \ldots, N_\Xi$ and adds the maximum error point in Ξ to $\mathcal{DB}_{N_p}$ to form $\mathcal{DB}_{N_p+1}$. The greedy algorithm repeats this process until the maximum error indicator among the points in Ξ is less than a threshold ϵ_{tol}.

In practice, there are two types of error indicators: 1) error bounds $\Delta(\boldsymbol{\mu})$ and 2) indicators based on the norm of the residual $\|\mathbf{r}(\mathbf{w})\|$. Error bounds $\Delta(\mathbf{w})$ rigorously satisfy

$$\|\mathbf{w}^\star - \mathbf{V}\mathbf{w}_r\| \leq \Delta(\mathbf{V}\mathbf{w}_r), \forall \mathbf{w}_r \in \mathbb{R}^k, \tag{13}$$

where $\mathbf{w}^\star$ denotes the PDE solution. Such error bounds have been derived for elliptic [32, 35], parabolic [21] and hyperbolic [22] PDEs, linear time invariant systems [23], and nonlinear eigenvalue problems [12]. However, these error bounds typically require the computation of an inf-sup constant and are not usually tight [5]. They are therefore currently limited to the aforementioned classes of equations. The norm of the residual $\mathbf{r}(\boldsymbol{\mu})$ is another popular alternative to error bounds [9, 10] in cases where such an error bound does not exist or is not a tight indicator of the error.

The requirement of computing $\varrho(\boldsymbol{\mu}; \mathcal{DB}_{N_p})$ for all $\boldsymbol{\mu} \in \Xi$ at each iteration N_p can be very expensive when the number of candidates N_Ξ is large. Two recent papers address this issue [31, 24]. The authors in [31] propose to use a surrogate model of error surface to find the point of the global maximum error indicator in $\mathcal{D}$. The authors in [24] propose an improved greedy algorithm by adopting the concept of the saturation assumption, which is widely used in the adaptive finite element mesh refinement algorithm. Alternative adaptive greedy algorithms are also proposed in the subsequent section.

Algorithm 1 Saturation assumption-based adaptive greedy algorithm for the ROM-interpolation model

Input: A candidate set $\Xi \subset \mathcal{D}$, the candidate set size N_Ξ, marginal factor γ, a tolerance $\epsilon_{tol} > 0$, an initial saturation constant $\widehat{\tau}_s$, the subset size N_Π
Output: A ROM database $\mathcal{DB}_{N_p}$

```
1: Choose an initial parameter value $\boldsymbol{\mu}_1 \in \Xi$, compute $\mathbf{A}_r(\boldsymbol{\mu}_1)$ and $\mathbf{b}_r(\boldsymbol{\mu}_1)$
2: Set $\mathcal{DB}_1 = \{(\boldsymbol{\mu}_1, \mathbf{A}_r(\boldsymbol{\mu}_1), \mathbf{b}_r(\boldsymbol{\mu}_1))\}$
3: Set $\varrho_{\text{profile}}^{\text{PREV}}(\boldsymbol{\mu}) = \varrho_{\text{profile}}(\boldsymbol{\mu}) = \infty$ for all $\boldsymbol{\mu} \in \Xi$ and $\varrho^c_{\max} = \infty$
4: while $\varrho^c_{\max} > \epsilon_{tol}$ do
5:   Generate a random subset $\Pi_{N_p}$ of $\Xi$
6:   Set $\varrho^c_{\max} = 1$ and $\tau_{\text{temp}}(j) = 0, j = 1, \dots, N_\Pi$
7:   for $\boldsymbol{\mu}_j \in \Pi_{N_p}, j = 1, \dots, N_\Pi$ do
8:     if $\widehat{\tau}_s \varrho_{\text{profile}}(\boldsymbol{\mu}_j) > \varrho^c_{\max}$ and $\widehat{\tau}_s \varrho_{\text{profile}}(\boldsymbol{\mu}_j) > \epsilon_{tol}$ then
9:       Set $\varrho_{\text{profile}}^{\text{PREV}}(\boldsymbol{\mu}_j) = \varrho_{\text{profile}}(\boldsymbol{\mu}_j)$
10:      Compute $\varrho(\boldsymbol{\mu}_j; \mathcal{DB}_{N_p})$, set $\varrho_{\text{profile}}(\boldsymbol{\mu}_j) = \varrho(\boldsymbol{\mu}_j; \mathcal{DB}_{N_p})$
11:      if $\varrho_{\text{profile}}^{\text{PREV}}(\boldsymbol{\mu}_j) \neq \infty$ and $\varrho_{\text{profile}}^{\text{PREV}}(\boldsymbol{\mu}_j) > \epsilon_{tol}$ then
12:        Set $\tau_{\text{temp}}(j) = \max(1, \gamma \frac{\varrho_{\text{profile}}(\boldsymbol{\mu}_j)}{\varrho_{\text{profile}}^{\text{PREV}}(\boldsymbol{\mu}_j)})$
13:      end if
14:      if $\varrho(\boldsymbol{\mu}_j; \mathcal{DB}_{N_p}) > \varrho^c_{\max}$ then
15:        Set $\varrho^c_{\max} = \varrho(\boldsymbol{\mu}_j; \mathcal{DB}_{N_p})$ and $\boldsymbol{\mu}_{N_p+1} = \boldsymbol{\mu}_j$
16:      end if
17:    end if
18:  end for
19:  if $\varrho^c_{\max} < \epsilon_{tol}$ then
20:    Perform a sanity check
21:  end if
22:  if $\max(\tau_{\text{temp}}) \geq 1$ then
23:    Set $\widehat{\tau}_s = \max(\tau_{\text{temp}})$
24:  end if
25:  Compute $\mathbf{A}_r(\boldsymbol{\mu}_{N_p+1})$ and $\mathbf{b}_r(\boldsymbol{\mu}_{N_p+1})$
26:  Set $\mathcal{DB}_{N_p+1} = \mathcal{DB}_{N_p} \cup \{(\boldsymbol{\mu}_{N_p+1}, \mathbf{A}_r(\boldsymbol{\mu}_{N_p+1}), \mathbf{b}_r(\boldsymbol{\mu}_{N_p+1}))\}$
27:  $N_p \leftarrow N_p + 1$
28: end while
```

4.2.2 Random greedy procedure

In order to speed up the process of the greedy algorithm, a random subset of Ξ can be evaluated at each iteration, instead of the whole candidate set Ξ. Let Π_{N_p} denote a subset of Ξ with size $N_\Pi \ll N_\Xi$ randomly selected from Ξ at Iteration N_p. The subset Π_{N_p} is updated in each greedy iteration in order to explore the parameter space $\mathcal{D}$. Additionally, only the points away from $\mathcal{DB}_{N_p}$ are included in Π_{N_p}. This condition can be reasoned from the fact that the true maximum error is likely to happen at the point away from $\mathcal{DB}_{N_p}$. The process is repeated until a convergence condition (i.e., the maximum error estimate $\varrho^c_{\max}$ is less than a desirable convergence tolerance ϵ_{tol}) is satisfied. If the convergence condition is satisfied at the end of the current greedy iteration. a sanity check is done by evaluating error indicators at points in a random subset Π_{N_p} of larger size. The sanity check is required to ensure that the database $\mathcal{DB}_{N_p}$ does not have any important missing points in $\mathcal{D}$. If the convergence condition $\varrho^c_{\max} < \epsilon_{tol}$ is still satisfied after the sanity check, then it returns $\mathcal{DB}_{N_p}$. If not, the point that gives the maximum error indicator in the sanity check is added to the database and the greedy procedure is continued.

4.2.3 Saturation constant assumption-based greedy procedure

The random greedy procedure can be further accelerated by applying the saturation assumption-based filtering proposed by [24]. It is detailed as follows.

Definition 1. ***Saturation Constant***
Let $\varrho(\boldsymbol{\mu};\mathcal{DB}_{N_p})$ denote an error indicator depending on a parameter $\boldsymbol{\mu}$ and a database $\mathcal{DB}_{N_p}$ with nested databases satisfying $\mathcal{DB}_{N_p} \subset \mathcal{DB}_{M_p}$ for all $1 \le N_p < M_p$. The ***Saturation Constant*** *$\tau_s > 0$ is defined as*

$$\tau_s = \underset{1\le N_p<M_p,\boldsymbol{\mu}\in\mathcal{D}}{sup} \ \frac{\varrho(\boldsymbol{\mu};\mathcal{DB}_{M_p})}{\varrho(\boldsymbol{\mu};\mathcal{DB}_{N_p})}. \tag{14}$$

Note the fact that $\varrho(\boldsymbol{\mu};\mathcal{DB}_{M_p}) \le \tau_s\varrho(\boldsymbol{\mu};\mathcal{DB}_{N_p})$ for all $1 \le N_p < M_p$ follows due to the definition of **Saturation Constant**. Note that $\tau_s < 1$ implies that $\varrho(\boldsymbol{\mu};\mathcal{DB}_{M_p}) < \varrho(\boldsymbol{\mu};\mathcal{DB}_{N_p})$ for all $1 \le N_p < M_p$. In other words, $\varrho(\boldsymbol{\mu};\cdot)$ strictly decreases as more points are included in the database. Similarly, $\tau_s = 1$ implies a monotone decrease in $\varrho(\boldsymbol{\mu};\cdot)$ as the number of points in the database increases. If $\tau_s > 1$, $\varrho(\boldsymbol{\mu};\cdot)$ may increase at some point $\boldsymbol{\mu} \in \mathcal{DB}$ for certain iterations of the greedy algorithm.

Assume that the saturation constant τ_s is known. At Iteration N_p, let $\varrho_{\text{profile}}(\boldsymbol{\mu}_c)$ denote the most recent available error estimate at $\boldsymbol{\mu}_c \in \Xi$: Let $\varrho^c_{\max}(\mathcal{DB}_{N_p})$ denote the maximum error estimate among all the previously computed error estimates at Iteration N_p, that is,

$$\varrho^c_{\max}(\mathcal{DB}_{N_p}) = \begin{cases} \max\limits_{j=1,\dots,c-1} \varrho(\boldsymbol{\mu}_j;\mathcal{DB}_{N_p}) & \text{for } 1 < c \le N_\Xi \\ 0 & \text{for } c = 1. \end{cases} \tag{15}$$

If $\tau_s\varrho_{\text{profile}}(\boldsymbol{\mu}_c) < \varrho^c_{\max}(\mathcal{DB}_{N_p})$, then $\varrho(\boldsymbol{\mu}_c;\mathcal{DB}_{N_p})$ is guaranteed to be less than $\varrho^c_{\max}(\mathcal{DB}_{N_p})$ due to the definition of the saturation constant. Thus it is not necessary to compute $\varrho(\boldsymbol{\mu}_c;\mathcal{DB}_{N_p})$ if $\tau_s\varrho_{\text{profile}}(\boldsymbol{\mu}_c) < \varrho^c_{\max}(\mathcal{DB}_{N_p})$. Hesthaven, et al. use this property to avoid computing error indicators at some points in Ξ and save computational time in their improved greedy algorithm [24].

However, the saturation constant is not known a priori in general except for special cases only. For example, if a global ROM is constructed without truncation for a symmetric coercive elliptic problem and the error is measured in the intrinsic energy norm, then the saturation constant τ_s is one. Hence, in general, the saturation assumption may not be satisfied. Nevertheless, it is still possible to use this concept to avoid a large number of error indicator evaluations as demonstrated below.

In the present work, the constant τ_s is one initially, then estimated and updated at each iteration of the greedy procedure. Based on the definition of the saturation constant (14), the following estimate for τ_s is possible at Iteration N_p:

$$\widehat{\tau}_s \approx \max_{\boldsymbol{\mu}_c\in\Xi,\varrho^{\text{PREV}}_{\text{profile}}(\boldsymbol{\mu}_c)>\epsilon_{tol}} \frac{\varrho_{\text{profile}}(\boldsymbol{\mu}_c)}{\varrho^{\text{PREV}}_{\text{profile}}(\boldsymbol{\mu}_c)}. \tag{16}$$

where $\varrho^{\text{PREV}}_{\text{profile}}(\boldsymbol{\mu}_c)$ is a preceeding available error estimate to $\varrho_{\text{profile}}(\boldsymbol{\mu}_c)$ at $\boldsymbol{\mu}_c$. The points with small error estimates can give a misleadingly large estimate $\widehat{\tau}_s$. Thus, the condition $\varrho^{\text{PREV}}_{\text{profile}}(\boldsymbol{\mu}_c) > \epsilon_{tol}$ is imposed in order to avoid those points with small error estimates in estimating τ_s.

A safety growth factor γ is then introduced because the approximation (16) is still a lower bound for τ_s. Algorithm 1 presents the saturation assumption-based adaptive greedy algorithm for the ROM-interpolation model. In Line 12 of the algorithm, $\tau_{\text{temp}}(j)$ is taken the maximum value among the pair $\left(1, \gamma\frac{\varrho_{\text{profile}}(\boldsymbol{\mu}_j)}{\varrho^{\text{PREV}}_{\text{profile}}(\boldsymbol{\mu}_j)}\right)$. It makes $\tau_s \ge 1$ throughout the greedy iterations. Similarly, as in the random greedy procedure, a sanity check without saturation-assumption filtering is also done after convergence of the greedy procedure.

Remark. The ROM-interpolation model relies on the interpolation scheme that is used to interpolate ROMs in $\mathcal{DB}$ (see Section 5). An over-fitting issue can cause a large fluctuation for some points in $\mathcal{D}$ even though several points are added to $\mathcal{DB}$. In order to prevent the over-fitting issue, it is recommended to use a smooth interpolation that is close to a linear interpolation. This can be accomplished, for example, by the multi-quadric radial basis function, which is defined in (17), with a low value of θ as depicted in Figure 3.

$$\phi(r) = \sqrt{r^2 + \theta^2} \tag{17}$$

The performance of the saturation-based adaptive greedy Algorithm 1 is compared both with a classical greedy algorithm and the surrogate-based greedy algorithm in Section 6.3.

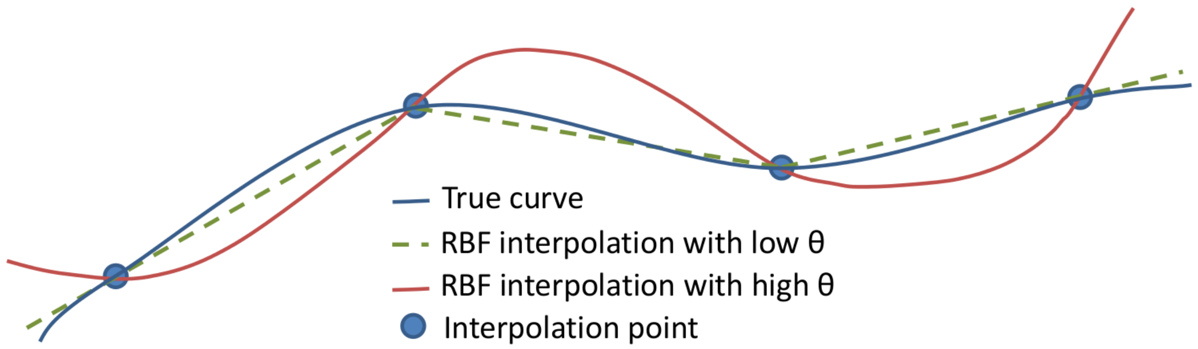


Figure 3: Illustration of overfitting problems and linear interpolation

5 Interpolation of parametric ROMs

A methodology of interpolating ROMs on matrix manifolds in [3, 1, 7] is introduced. It enables the computation of a reduced-order model at any point in the parameter space in real time. Section 5.1 presents a brief overview of the consistent interpolation of the local ROMs on matrix manifolds. The sensitivity of the interpolation on matrix manifolds is introduced for gradient-based optimization algorithms in Section 5.2. Section 5.3 addresses the computation of residual-based error estimates in the context of ROM interpolation.

5.1 Consistent interpolation of local ROMs on matrix manifolds

The interpolation scheme has two major steps: 1) Identification of congruence transformations and 2) Interpolation on matrix manifolds. The first step is required because ROMs that are constructed for different parameter values (e.g., $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$) are usually not expressed in a consistent set of generalized coordinates [3]. In other words, the corresponding ROMs $\mathbf{A}_r(\boldsymbol{\mu}_1)$ and $\mathbf{A}_r(\boldsymbol{\mu}_2)$ (and similarly $\mathbf{b}_r(\boldsymbol{\mu}_1)$ and $\mathbf{b}_r(\boldsymbol{\mu}_2)$) are not expressed in a consistent way so that they cannot be compared directly. Fortunately, congruent transformation of inconsistent ROMs can be defined by noticing that there are equivalence classes of ROBs for a given subspace $\mathcal{S}(\boldsymbol{\mu}) = \text{ range}(\mathbf{V}(\boldsymbol{\mu}))$:

$$\{\widetilde{\mathbf{V}}(\boldsymbol{\mu}) = \mathbf{V}(\boldsymbol{\mu})\mathbf{Q}\}, \tag{18}$$

where $\mathbf{Q} \in \mathbb{R}^{k\times k}$ is an orthogonal matrix. Considering for simplicity the case of Galerkin projection, the ROM defined in (9) is transformed as

$$\widetilde{\mathbf{A}}_r(\boldsymbol{\mu}) = \mathbf{Q}^T\mathbf{A}_r(\boldsymbol{\mu})\mathbf{Q} \qquad \widetilde{\mathbf{b}}_r(\boldsymbol{\mu}) = \mathbf{Q}^T\mathbf{b}_r(\boldsymbol{\mu}). \tag{19}$$

Therefore, an equivalence class of ROMs can be defined as

$$c(\mathbf{A}_r, \mathbf{b}_r) = \{\mathbf{Q}^T\mathbf{A}_r\mathbf{Q}, \mathbf{Q}^T\mathbf{b}_r | \mathbf{Q} \in O(k)\}, \tag{20}$$

where $O(k)$ denotes the set of orthogonal matrices of size k.

Optimal transformations are then computed as follows [3]. Let $\mathbf{V}(\boldsymbol{\mu}_1), \mathbf{V}(\boldsymbol{\mu}_2), \ldots,$ and $\mathbf{V}(\boldsymbol{\mu}_{N_p})$ denote the ROBs associated with each ROM. If $\boldsymbol{\mu}_1$ is a reference parameter value, then in the first step, the following series of orthogonal Procrustes problems are solved in order to find congruence matrices that transform each ROB $\mathbf{V}(\boldsymbol{\mu}_c)$, $c = 1, \ldots, N_p$ into a ROB consistent with the reference basis $\mathbf{V}(\boldsymbol{\mu}_1)$:

$$\begin{aligned} \underset{\mathbf{Q}_c \in \mathbb{R}^{k\times k}}{\text{minimize}} \quad & \|\mathbf{V}(\boldsymbol{\mu}_c)\mathbf{Q}_c - \mathbf{V}(\boldsymbol{\mu}_1)\|_F^2 \\ \text{subject to} \quad & \mathbf{Q}_c^T\mathbf{Q}_c = \mathbf{I}_k, \qquad c = 1, \ldots, N_p. \end{aligned} \tag{21}$$

The optimal solution to (21) is analytically given by

$$\mathbf{Q}_c^\star = \mathbf{U}_c\mathbf{Z}_c^T, \tag{22}$$

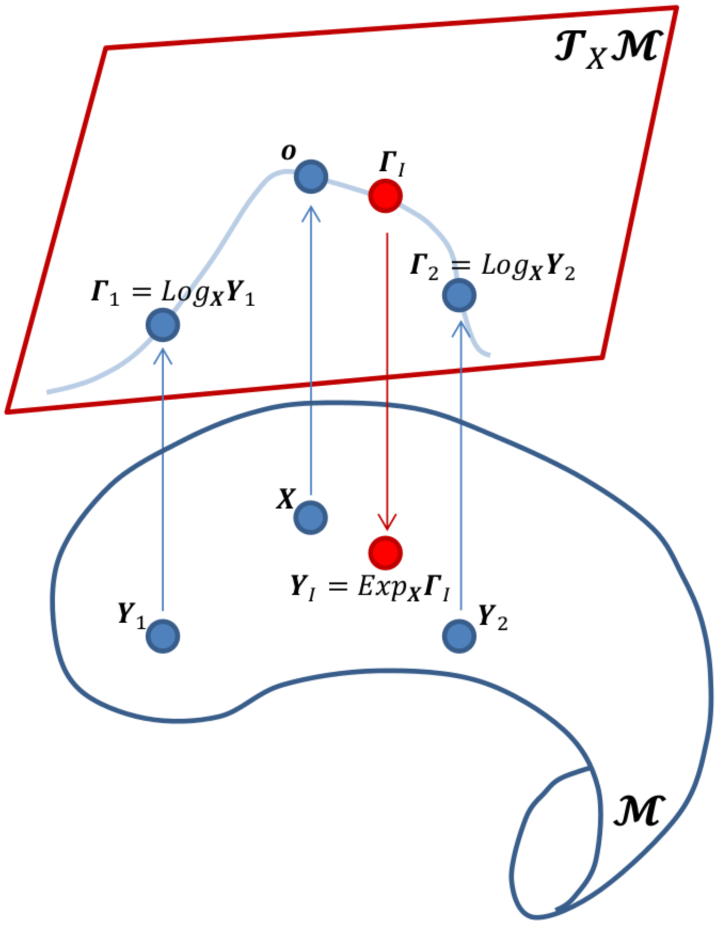


Figure 4: Schematic representation of logarithm and exponential mappings and application to interpolation

where $\mathbf{U}_c\boldsymbol{\Sigma}_c\mathbf{Z}_c^T$ is a singular value decomposition of $\mathbf{V}(\boldsymbol{\mu}_c)^T\mathbf{V}(\boldsymbol{\mu}_1)$ (e.g., see [20]). Once the transformation matrices $\mathbf{Q}_c^\star$ are computed, a set of rotated ROMs

$$\{\widetilde{\mathbf{A}}_r(\boldsymbol{\mu}_c), \widetilde{\mathbf{b}}_r(\boldsymbol{\mu}_c)\} = \{(\mathbf{Q}_c^\star)^T\mathbf{A}_r(\boldsymbol{\mu}_c)\mathbf{Q}_c^\star, (\mathbf{Q}_c^\star)^T\mathbf{b}(\boldsymbol{\mu}_c)\} \tag{23}$$

can be obtained via (19) for $c = 1, \dots, N_p$.

In the second step, interpolation of the consistent ROMs $\{\widetilde{\mathbf{A}}_r(\boldsymbol{\mu}_c), \widetilde{\mathbf{b}}_r(\boldsymbol{\mu}_c)\}$, $c = 1, \dots, N_p$ is performed on matrix manifolds leading to an approximation of $\{\widetilde{\mathbf{A}}_r(\boldsymbol{\mu}^\star), \widetilde{\mathbf{b}}_r(\boldsymbol{\mu}^\star)\}$ for a new value $\boldsymbol{\mu}^\star \in \mathcal{D}$. Interpolation on matrix manifolds is necessary in order to preserve any characteristics that a ROM operator $\widetilde{\mathbf{A}}_r(\boldsymbol{\mu}_c)$ might have (e.g., orthogonality, non-singularity, symmetry, or positive-definiteness). Interpolation on matrix manifolds can be divided into three sub-steps:

1. logarithm mappings of the consistent ROMs to a linear tangent space
2. interpolating mapped data in the linear tangent space
3. exponential mappings of the interpolated quantity from the linear tangent space back to the original manifold.

More specifically, let $\mathcal{M}$ be a manifold in $\mathbb{R}^{M\times N}$ whose elements are characterized by properties such as orthogonality, non-singularity, symmetry, or positive-definiteness. Let $\mathbf{X} = \widetilde{\mathbf{A}}_r(\boldsymbol{\mu}_1) \in \mathcal{M}$ be a reference

Table 1: Exponential and logarithm mappings for the matrix manifolds of interest.

Manifold	$\mathcal{R}^{M\times N}$	Nonsingular matrices	SPD matrices
$\mathrm{Log}_{\mathbf{X}}(\mathbf{Y})$	$\mathbf{Y}-\mathbf{X}$	$\log(\mathbf{Y}\mathbf{X}^{-1})$	$\log(\mathbf{X}^{-1/2}\mathbf{Y}\mathbf{X}^{-1/2})$
$\mathrm{Exp}_{\mathbf{X}}(\boldsymbol{\Gamma})$	$\mathbf{X}+\boldsymbol{\Gamma}$	$\exp(\boldsymbol{\Gamma})\mathbf{X}$	$\mathbf{X}^{1/2}\exp(\boldsymbol{\Gamma})\mathbf{X}^{1/2}$

Algorithm 2 Interpolation on a manifold $\mathcal{M}$

```
Input: $N_p$ matrices $\mathbf{Y}_1, \dots, \mathbf{Y}_{N_p}$ belonging to $\mathcal{M}$
Output: Interpolated matrix $\mathbf{Y}^\star$
1: Set a reference point $\mathbf{X} = \mathbf{Y}_1$ (e.g., The interpolation process takes place in the linear space $\mathcal{T}_{\mathbf{Y}_1}\mathcal{M}$).
2: for $c = 1, \dots, N_p$ do
3:     Compute $\boldsymbol{\Gamma}_c = \mathrm{Log}_{\mathbf{X}}(\mathbf{Y}_c)$
4: end for
5: Interpolate each entry of the matrices $\boldsymbol{\Gamma}_c$, $c = 1, \dots, N_p$, independently to obtain $\boldsymbol{\Gamma}^\star$
6: Compute $\mathbf{Y}(\boldsymbol{\mu}^\star) = \mathbf{Y}^\star = \mathrm{Exp}_{\mathbf{Y}_1}(\boldsymbol{\Gamma}^\star)$
```

point and $\mathbf{Y}_c = \widetilde{\mathbf{A}}_r(\boldsymbol{\mu}_c) \in \mathcal{M}$ for $c = 1, \dots, N_p$ an element in the neighborhood of $\mathbf{X}$. Let also $\boldsymbol{\Gamma}$ be an element of the tangent space $\mathcal{T}_{\mathbf{X}}\mathcal{M}$ at $\mathbf{X}$. The logarithm mapping $\mathrm{Log}_{\mathbf{X}}$ defines a mapping from an element in a neighborhood of $\mathbf{X}$ (i.e., $\mathcal{N}(\mathbf{X})$) to an element in the tangent space $\mathcal{T}_{\mathbf{X}}\mathcal{M}$. On the other hand, the exponential mapping $\mathrm{Exp}_{\mathbf{X}}$ defines a mapping from the tangent space $\mathcal{T}_{\mathbf{X}}\mathcal{M}$ to $\mathcal{M}$. The neighborhood $\mathcal{N}(\mathbf{X})$ is identified by the property that for any element $\mathbf{Y} \in \mathcal{N}(\mathbf{X})$, the equation $\mathrm{Exp}_{\mathbf{X}}(\boldsymbol{\Gamma}) = \mathbf{Y}$ has a unique solution $\boldsymbol{\Gamma}$ satisfying $\mathrm{Log}_{\mathbf{X}}(\mathbf{Y}) = \boldsymbol{\Gamma}$.

The tangent space $\mathcal{T}_{\mathbf{X}}\mathcal{M}$ is a vector space, so it is easier to apply any interpolation scheme in $\mathcal{T}_{\mathbf{X}}\mathcal{M}$ than in $\mathcal{M}$. Let $\boldsymbol{\Gamma}_c = \mathrm{Log}_{\mathbf{X}}(\mathbf{Y}_c) \in \mathcal{T}_{\mathbf{X}}\mathcal{M}$ for $c = 1, \dots, N_p$ and $\mathcal{I}(\boldsymbol{\mu}^\star; \{\boldsymbol{\Gamma}_c\}_{c=1}^{N_p})$ be an interpolation operator that takes a set of N_p distinct elements $\boldsymbol{\Gamma}_c$, $c = 1, \dots, N_p$ as inputs and returns an interpolant $\boldsymbol{\Gamma}^\star \in \mathcal{T}_{\mathbf{X}}(\mathcal{M})$. Then the exponential mapping $\mathrm{Exp}_{\mathbf{X}}(\boldsymbol{\Gamma}^\star)$ returns the element $\mathbf{Y}^\star \in \mathcal{M}$, which completes the procedure of the interpolation on matrix manifolds. The procedure of the interpolation in matrix manifolds can be compactly expressed in the following equation:

$$\mathbf{Y}^\star = \mathbf{Y}(\boldsymbol{\mu}^\star) = \mathrm{Exp}_{\mathbf{X}}\left[\mathcal{I}\left(\boldsymbol{\mu}^\star; \{\mathrm{Log}_{\mathbf{X}}(\mathbf{Y}_c)\}_{c=1}^{N_p}\right)\right]. \tag{24}$$

Because the exponential mapping brings an element in $\mathcal{T}_{\mathbf{X}}\mathcal{M}$ back to a point in $\mathcal{M}$, the interpolation scheme on matrix manifolds described above produces an interpolant (e.g., $\widetilde{\mathbf{A}}_r(\boldsymbol{\mu}^\star)$) that has the same properties as the interpolated points $\widetilde{\mathbf{A}}_r(\boldsymbol{\mu}_c), c = 1, \dots, N_p$ (e.g., orthogonality, non-singularity, and positive-definiteness). Figure 4 graphically depicts the exponential and logarithm mappings in a manifold and interpolation in the linear tangent space. Table 1 shows the expressions of exponential and logarithm mappings for various manifolds of matrices such as real matrices, non-singular matrices, symmetric positive-definite matrices. Finally, Algorithm 2 summarizes the procedure of the interpolation on matrix manifolds. For more detailed descriptions of the interpolation of ROMs and ROBs on matrix manifolds, see [2] and [3].

5.2 Online computation of sensitivities

Using analytical gradients instead of a finite difference approximations can speed up the convergence of gradient-based optimization algorithms. Furthermore finite difference approximations in the context of PDE-constrained optimization can be very expensive due to the requirement of solving the underlying PDE $N_\mu + 1$ times if forward or backward difference is used and $2N_\mu$ times if central difference is used. To alleviate that cost, analytical sensitivities of the interpolated ROMs are derived in this section.

As shown in Section 2, $\frac{\partial \widetilde{\mathbf{A}}}{\partial \mu_i}$ and $\frac{\partial \widetilde{\mathbf{b}}}{\partial \mu_i}$ need to be computed to obtain gradients at a given parameter $\boldsymbol{\mu}^\star$. In the context of optimization where the high-dimensional model is replaced by a ROM as in (9) and (10), the sensitivities of the reduced operators $\frac{\partial \widetilde{\mathbf{A}}_r}{\partial \mu_i}$ and $\frac{\partial \widetilde{\mathbf{b}}_r}{\partial \mu_i}$ need to be computed instead. In

Table 2: Sensitivities of exponential mappings for the matrix manifolds of interest.

Manifold	$\mathcal{R}^{M\times N}$	Nonsingular matrices	SPD matrices
$\frac{\partial \mathrm{Exp}_{\mathbf{X}}(\boldsymbol{\Gamma})}{\partial \mu_i}$	$\frac{\partial \boldsymbol{\Gamma}}{\partial \mu_i}$	$\frac{\partial \exp(\boldsymbol{\Gamma})}{\partial \mu_i}\mathbf{X}$	$\mathbf{X}^{1/2}\frac{\partial \exp(\boldsymbol{\Gamma})}{\partial \mu_i}\mathbf{X}^{1/2}$

the proposed methodology, $\widetilde{\mathbf{A}}_r$ is computed by interpolation within a database of consistent ROMs as compactly described in Eq. (24). Note that the operators $\{\mathrm{Log}_{\mathbf{X}}(\mathbf{Y}_c)\}_{c=1}^{N_p}$ are independent of $\boldsymbol{\mu}$. Thus, the derivatives of $\mathbf{Y}^\star(\boldsymbol{\mu})$ with respect to $\boldsymbol{\mu}$ become

$$\frac{d\mathbf{Y}^\star}{d\boldsymbol{\mu}} = \frac{d\mathbf{Y}}{d\boldsymbol{\mu}}(\boldsymbol{\mu}^\star) = \frac{d\mathrm{Exp}_{\mathbf{X}}\mathcal{I}}{d\mathcal{I}} \cdot \frac{d\mathcal{I}}{d\boldsymbol{\mu}}\left(\boldsymbol{\mu}^\star; \{\mathrm{Log}_{\mathbf{X}}(\mathbf{Y}_c)\}_{c=1}^{N_p}\right). \tag{25}$$

The derivatives of the exponential mapping $\frac{\partial \mathrm{Exp}_{\mathbf{X}}\boldsymbol{\Gamma}}{\partial \mu_i}$ associated with several matrix manifolds are provided in Table 2. Note that the derivatives of the matrix exponential (e.g., $\frac{\partial \exp(\boldsymbol{\Gamma})}{\partial \mu_i}$) are required both for the manifolds of nonsingular matrices and SPD matrices. In order to obtain the derivatives of the matrix exponential $\frac{\partial \exp(\boldsymbol{\Gamma})}{\partial \mu_i}$, one can first define a matrix

$$\mathbf{B} = \begin{bmatrix} \boldsymbol{\Gamma} & \frac{\partial \boldsymbol{\Gamma}}{\partial \mu_i} \\ \mathbf{0} & \boldsymbol{\Gamma} \end{bmatrix} \in \mathbb{R}^{2k\times 2k}. \tag{26}$$

Following [30], the exponential matrix of $\mathbf{B}$ becomes

$$\exp(\mathbf{B}) = \begin{bmatrix} \exp(\boldsymbol{\Gamma}) & \frac{\partial \exp(\boldsymbol{\Gamma})}{\partial \mu_i} \\ \mathbf{0} & \exp(\boldsymbol{\Gamma}) \end{bmatrix}. \tag{27}$$

Hence, the derivative of the matrix exponential $\frac{\partial \exp(\boldsymbol{\Gamma})}{\partial \mu_i}$ can be simply extracted as the (1,2)-block of $\exp(\mathbf{B})$. Note that the computation of $\exp(\mathbf{B})$ is inexpensive as it operates on a reduced size matrix of dimension $2k$.

As for $\widetilde{\mathbf{b}}_r$ and $\frac{\partial \widetilde{\mathbf{b}}_r}{\partial \mu_i}$, one can apply the same interpolation technique described above on the matrix manifold $\mathbb{R}^{k\times 1}$. Once $\frac{\partial \widetilde{\mathbf{A}}_r}{\partial \mu_i}$ and $\frac{\partial \widetilde{\mathbf{b}}_r}{\partial \mu_i}$ are computed, the sensitivity $\frac{dq}{d\mu_i}$ can be computed by the chain rule:

$$\frac{dq}{d\mu_i}(\mathbf{w}_r(\boldsymbol{\mu}), \boldsymbol{\mu}) = \frac{\partial q}{\partial \mu_i}(\mathbf{w}_r(\boldsymbol{\mu}), \boldsymbol{\mu}) + \left(\frac{\partial q}{\partial \mathbf{w}_r}(\mathbf{w}_r(\boldsymbol{\mu}), \boldsymbol{\mu})\right)\widetilde{\mathbf{A}}_r(\boldsymbol{\mu})^{-1}\left(\frac{\partial \widetilde{\mathbf{b}}_r}{\partial \mu_i}(\boldsymbol{\mu}) - \frac{\partial \widetilde{\mathbf{A}}_r}{\partial \mu_i}(\boldsymbol{\mu})\mathbf{w}_r(\boldsymbol{\mu})\right). \tag{28}$$

5.3 Computation of residual-based error indicators

When residual-based error indicators are used in the greedy algorithm, the residual is computed as

$$\mathbf{r}(\boldsymbol{\mu}, \mathbf{w}_r) = \mathbf{b}(\boldsymbol{\mu}) - \mathbf{A}(\boldsymbol{\mu})\mathbf{V}(\boldsymbol{\mu})\mathbf{w}_r, \tag{29}$$

where $\mathbf{w}_r$ is the solution to $\widetilde{\mathbf{A}}_r(\boldsymbol{\mu})\mathbf{w}_r = \widetilde{\mathbf{b}}_r(\boldsymbol{\mu})$ and $\mathbf{V}(\boldsymbol{\mu})$ is the ROB. In the present context, if $\boldsymbol{\mu} \notin \mathcal{DB}$, only the reduced order models (i.e., $\widetilde{\mathbf{A}}_r$ and $\widetilde{\mathbf{b}}_r$) are interpolated and $\mathbf{V}(\boldsymbol{\mu})$ is not available. It is possible to obtain the ROB $\mathbf{V}(\boldsymbol{\mu})$ by interpolating the ROBs $\mathbf{V}(\boldsymbol{\mu}_c)$, $c = 1, \dots, N_p$ on the Grassmannian manifold [2]. However, the interpolation on the Grassmannian manifold is much more expensive than the interpolation of the ROMs because it requires the thin SVD of large scale matrices. One way of avoiding the interpolation of the ROBs is to replace $\mathbf{V}(\boldsymbol{\mu})$ with $\mathbf{V}(\boldsymbol{\mu}_r)$ where $\boldsymbol{\mu}_r \in \mathcal{DB}$ is the closest point to $\boldsymbol{\mu}$ (e.g., $\boldsymbol{\mu}_r = \underset{\boldsymbol{\mu}_i \in \mathcal{DB}}{\mathrm{argmin}}\|\boldsymbol{\mu} - \boldsymbol{\mu}_i\|$). The choice of $\boldsymbol{\mu}_r$ is illustrated in Figure 5 for a case where the database $\mathcal{DB}$ has five points in $\mathbb{R}^2$. For example, the interpolants at $\boldsymbol{\mu}^*$ uses the ROB at $\boldsymbol{\mu}_1$ to compute a residual

$$\mathbf{r}(\boldsymbol{\mu}^*, \mathbf{w}_r) = \mathbf{b}(\boldsymbol{\mu}^*) - \mathbf{A}(\boldsymbol{\mu}^*)\mathbf{V}(\boldsymbol{\mu}_1)\mathbf{w}_r. \tag{30}$$

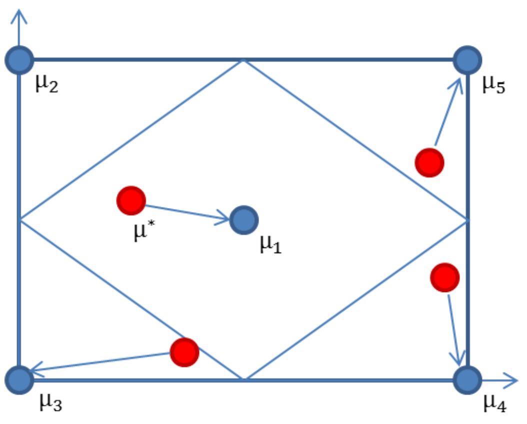


Figure 5: ROB choice for residual evaluation

6 Design optimization of a wing under aeroelastic constraints

The following design optimization problem of a wing under aeroelastic constraints is considered to demonstrate the performance of the proposed ROM database strategy:

$$\begin{aligned} \underset{\boldsymbol{\mu}\in\mathcal{D}}{\text{minimize}} \quad & \frac{L(\boldsymbol{\mu})}{D(\boldsymbol{\mu})} \\ \text{subject to} \quad & \sigma_{\text{VM}}(\boldsymbol{\mu}) \le \sigma_{\text{upper}} \\ & \boldsymbol{\zeta}(\boldsymbol{\mu}) \ge \boldsymbol{\zeta}_{\text{lower}} \\ & W(\boldsymbol{\mu}) \le W_{\text{upper}} \\ & \boldsymbol{\mu}_{\text{lower}} \le \boldsymbol{\mu} \le \boldsymbol{\mu}_{\text{upper}}, \end{aligned} \tag{31}$$

where $\boldsymbol{\mu}$ is a design parameter vector belonging to a desing space $\mathcal{D}$, $L(\boldsymbol{\mu})$ and $D(\boldsymbol{\mu})$ are the lift and drag, respectively, and $W(\boldsymbol{\mu})$ is the weight of the wing. The von Mises stress σ_{VM} of the wing at the steady state is constrained not to exceed an yield stress σ_{upper}. The damping ratio vector $\boldsymbol{\zeta}(\boldsymbol{\mu})$ is not allowed to be below a lower bound $\boldsymbol{\zeta}_{\text{lower}} > \mathbf{0}$ to avoid flutter. Bound constraints on $\boldsymbol{\mu}$ are introduced to avoid unrealistic designs. Finally, two types of design parameters are considered in $\boldsymbol{\mu} = (\boldsymbol{\mu}_s, \boldsymbol{\mu}_m)$: external shape parameters $\boldsymbol{\mu}_s \in \mathbb{R}^{p_s}$ and structural parameters $\boldsymbol{\mu}_m \in \mathbb{R}^{p_m}$. The external shape parameters $\boldsymbol{\mu}_s$ affects both the shape of the structure and the fluid domain through its interface with the structure. The vector of parameters $\boldsymbol{\mu}_m \in \mathbb{R}^{p_m}$ contains the material properties of the structural system as well as internal shape parameters associated with the "dry" elements of the structure that are not in contact with the external flow.

The constraints can be grouped into two sets:

- Static aeroelastic constraints that are computed using a HDM that derives from a three-field formulation [28]. These constraints include the weight, lift-drag ratio, and von Mises stress.
- Dynamic aeroelastic constraints (flutter) computed by a linearized HDM that derives from a three-field formulation linearized around an equilibrium state [27]. The proposed ROM database strategy is used to alleviate the large computational cost associated with the HDM for the flutter constraints.

The damping ratio $\boldsymbol{\zeta}$ and its sensitivities $\frac{d\boldsymbol{\zeta}}{d\boldsymbol{\mu}}$ are then computed via the ROM interpolation technique described in Section 5.2.

Section 6.1 presents the linearized aeroelastic equation and its model reduction. The section also shows the derivation of the aeroelastic damping ratios and their interpolations. The computational results regarding solutions of the optimization problem (31) are presented in Section 6.2.

6.1 Linearized CFD-based fluid-structure interaction and model reduction

A CFD-based, nonlinear, high-fidelity aeroelastic system can be described by a three-field Arbitrary Lagrangian Eulerian (ALE) formulation that considers the moving mesh as a pseudo-structural system and can handle large deformations [17]. After semi-discretization in space, linearization of the semi-discrete equations about an equilibrium state [27], and the elimination of the fluid mesh position, the following system of linear ODEs is obtained:

$$\begin{aligned}\mathbf{A}(\boldsymbol{\mu})\dot{\mathbf{w}} + \mathbf{H}(\boldsymbol{\mu})\mathbf{w} + \mathbf{R}(\boldsymbol{\mu})\dot{\mathbf{u}} + \mathbf{G}(\boldsymbol{\mu})\mathbf{u} &= \mathbf{0} &(32)\\ \mathbf{M}(\boldsymbol{\mu})\ddot{\mathbf{u}} + \mathbf{D}(\boldsymbol{\mu})\dot{\mathbf{u}} + \mathbf{K}(\boldsymbol{\mu})\mathbf{u} &= \mathbf{P}(\boldsymbol{\mu})\mathbf{w}. &(33)\end{aligned}$$

where a dot denotes a derivative with respect to time t, $\mathbf{A} \in \mathbb{R}^{N_f \times N_f}$ the diagonal matrix of cell volumes in the fluid mesh, and N_f the dimension of the semi-discretized fluid subsystem. The conservative fluid state vector is denoted as $\mathbf{w}(t) \in \mathbb{R}^{N_f}$ and $\mathbf{u}(t) \in \mathbb{R}^{N_s}$ is the vector of structural displacements of dimension N_s. The Jacobians of the numerical fluxes with respect to $\mathbf{w}$ and $\mathbf{x}$ are denoted as $\mathbf{H} \in \mathbb{R}^{N_f \times N_f}$ and $\mathbf{G} \in \mathbb{R}^{N_f \times N_x}$, respectively. The matrix $\mathbf{R} \in \mathbb{R}^{N_f \times N_x}$ appears due to the linearization of the underlying HDM with respect to the fluid mesh velocity. Both $\mathbf{G}$ and $\mathbf{R}$ are coupling terms. The mass matrix $\mathbf{M} \in \mathbb{R}^{N_s \times N_s}$ is associated with the Finite Element (FE) discretization of the structural subsystem and $\mathbf{D} \in \mathbb{R}^{N_s \times N_s}$ and $\mathbf{K} \in \mathbb{R}^{N_s \times N_s}$ are respectively the FE damping and stiffness matrices. The Jacobian of the aerodynamic forces $\mathbf{P} \in \mathbb{R}^{N_s \times N_f}$ acts on the surface of the structure with respect to $\mathbf{w}$.

First, the dimensionality of the structural subsystem is reduced using modal truncation. A ROB $\mathbf{X}(\boldsymbol{\mu}) \in \mathbb{R}^{N_s \times k_s}$ is constructed using the first k_s modes of the structural subsystem and Eq. (33) reduced by Galerkin projection onto $\mathbf{X}(\boldsymbol{\mu})$. This results in an approximation of $\mathbf{u}(t)$ as

$$\mathbf{u}(t) \approx \mathbf{X}(\boldsymbol{\mu})\mathbf{u}_r(t), \tag{34}$$

where the ROB $\mathbf{X}$ depends on the parameters $\boldsymbol{\mu} = (\boldsymbol{\mu}_s, \boldsymbol{\mu}_m) \in \mathbb{R}^{p_s+p_m}$ and $\mathbf{u}_r \in \mathbb{R}^{k_s}$ is a vector of k_s generalized coordinates associated with the modal approximation. The projection of (33) results in a set of k_s equations

$$\ddot{\mathbf{u}}_r + \mathbf{D}_r(\boldsymbol{\mu})\dot{\mathbf{u}}_r + \boldsymbol{\Omega}_r^2(\boldsymbol{\mu})\mathbf{u}_r = \mathbf{X}(\boldsymbol{\mu})^T\mathbf{P}(\boldsymbol{\mu})\mathbf{w}, \tag{35}$$

where $\mathbf{D}_r(\boldsymbol{\mu}) = \mathbf{X}(\boldsymbol{\mu})^T\mathbf{D}(\boldsymbol{\mu})\mathbf{X}(\boldsymbol{\mu}) \in \mathbb{R}^{k_s \times k_s}$ and $\boldsymbol{\Omega}_r^2 \in \mathbb{R}^{k_s \times k_s}$ is the diagonal matrix of eigenvalues associated with the k_s eigenmodes.

The dimensionality of the fluid subsystem is then reduced by POD in the frequency domain [25]. A ROB $\mathbf{V}(\boldsymbol{\mu}) \in \mathbb{R}^{N_f \times k_f}$ is built and the state vector $\mathbf{w}$ approximated as

$$\mathbf{w}(t) \approx \mathbf{V}(\boldsymbol{\mu})\mathbf{w}_r(t), \tag{36}$$

where $\mathbf{w}_r$ is a vector of k_f generalized coordinates associated with $\mathbf{V}(\boldsymbol{\mu})$. The procedure of constructing $\mathbf{V}(\boldsymbol{\mu})$ is outlined in Algorithm 3. Note that the orthogonality condition $\mathbf{V}(\boldsymbol{\mu})^T\mathbf{A}(\boldsymbol{\mu})\mathbf{V}(\boldsymbol{\mu}) = \mathbf{I}_{k_f}$ is satisfied [4]. Galerkin projection of (32) using $\mathbf{V}(\boldsymbol{\mu})$ results in the set of reduced coupled linear ODEs:

$$\begin{aligned}\dot{\mathbf{w}}_r + \mathbf{H}_r(\boldsymbol{\mu})\mathbf{w}_r + \mathbf{R}_r(\boldsymbol{\mu})\dot{\mathbf{u}}_r + \mathbf{G}_r(\boldsymbol{\mu})\mathbf{u}_r &= \mathbf{0} &(37)\\ \ddot{\mathbf{u}}_r + \mathbf{D}_r(\boldsymbol{\mu})\dot{\mathbf{u}}_r + \boldsymbol{\Omega}_r^2(\boldsymbol{\mu})\mathbf{u}_r - \mathbf{P}_r(\boldsymbol{\mu})\mathbf{w}_r &= \mathbf{0}, &(38)\end{aligned}$$

where $\mathbf{H}_r(\boldsymbol{\mu}) = \mathbf{V}(\boldsymbol{\mu})^T\mathbf{H}(\boldsymbol{\mu}_s)\mathbf{V}(\boldsymbol{\mu})$ and the coupling matrices are $\mathbf{R}_r(\boldsymbol{\mu}) = \mathbf{V}(\boldsymbol{\mu})^T\mathbf{R}(\boldsymbol{\mu}_s)\mathbf{X}(\boldsymbol{\mu})$ and $\mathbf{P}_r(\boldsymbol{\mu}) = \mathbf{X}(\boldsymbol{\mu})^T\mathbf{P}\mathbf{V}(\boldsymbol{\mu})$. In compact form, the system (37)–(38) can be written as

$$\dot{\mathbf{q}} = \mathbf{N}(\boldsymbol{\mu})\mathbf{q}, \tag{39}$$

where $\mathbf{q}(t) = \begin{bmatrix} \mathbf{w}_r(t) \\ \dot{\mathbf{u}}_r(t) \\ \mathbf{u}_r(t) \end{bmatrix} \in \mathbb{R}^{k_f+2k_s}$ and

$$\mathbf{N}(\boldsymbol{\mu}) = \begin{bmatrix} -\mathbf{H}_r(\boldsymbol{\mu}) & -\mathbf{R}_r(\boldsymbol{\mu}) & -\mathbf{G}_r(\boldsymbol{\mu}) \\ \mathbf{P}_r(\boldsymbol{\mu}) & -\mathbf{D}_r(\boldsymbol{\mu}) & -\boldsymbol{\Omega}_r^2(\boldsymbol{\mu}) \\ \mathbf{0} & \mathbf{I}_{k_s} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{(k_f+2k_s)\times(k_f+2k_s)}. \tag{40}$$

Algorithm 3 Construction of a fluid ROB for the aeroelastic system

```
Input: Parameter $\boldsymbol{\mu} = (\boldsymbol{\mu}_s, \boldsymbol{\mu}_m) \in \mathbb{R}^{p_s+p_m}$, frequency sampling $\{\xi_j\}_{j=1}^{N_\xi}$, eigenmodes $\mathbf{X}(\boldsymbol{\mu}) \in \mathbb{R}^{N_s \times k_s}$, ROB dimension $k_f$
Output: ROB $\mathbf{V}(\boldsymbol{\mu})$
1: Compute $\mathbf{A} = \mathbf{A}(\boldsymbol{\mu})$, $\mathbf{H} = \mathbf{H}(\boldsymbol{\mu})$, $\mathbf{R} = \mathbf{R}(\boldsymbol{\mu})$, and $\mathbf{G} = \mathbf{G}(\boldsymbol{\mu})$
2: for $i = 1, \cdots, k_s$ do
3:     for $l = 1, \cdots, N_\xi$ do
4:         Solve the linear system $(j\xi_l\mathbf{A} + \mathbf{H})\mathbf{w}_{i,l} = -(j\xi_l\mathbf{R} + \mathbf{G})\mathbf{x}_i$, where $\mathbf{x}_i$ is the $i$-th vector in $\mathbf{X}$
5:     end for
6: end for
7: Construct the complex-valued snapshot matrix $\mathbf{W} = [\mathbf{w}_{1,1}, \cdots, \mathbf{w}_{k_s,N_\omega}]$
8: Compute the singular value decomposition: $\mathbf{A}^{\frac{1}{2}}[\mathrm{Re}(\mathbf{W}), \mathrm{Im}(\mathbf{W})] = \mathbf{U}\boldsymbol{\Sigma}\mathbf{Z}^T$
9: Retain the first $k_f$ left singular vectors $\mathbf{V}(\boldsymbol{\mu}) = \mathbf{A}^{-\frac{1}{2}}\mathbf{U}_{k_f}$
```

For simplicity, and without loss of generality, the case of an undamped structure is considered in the remainder of this paper, that is $\mathbf{D}_r(\boldsymbol{\mu}) = \mathbf{0}$. For a given parameter vector $\boldsymbol{\mu}$ and altitude h, structural eigenvalues $\{\lambda_j(\boldsymbol{\mu})\}_{j=1}^{2k_s}$ can be extracted from the the eigen-decomposition of $\mathbf{N}(\boldsymbol{\mu})$ in (40):

$$\mathbf{N}(\boldsymbol{\mu})\hat{\mathbf{q}}_j(\boldsymbol{\mu}) = \lambda_j(\boldsymbol{\mu})\hat{\mathbf{q}}_j(\boldsymbol{\mu}), \qquad j = 1, \ldots, 2k_s + k_f. \tag{41}$$

An algorithm for efficiently extracting the $2k_s$ structural eigenvalues of $\mathbf{N}(\boldsymbol{\mu})$ is introduced in [6]. It proceeds by extracting structural eigenvalues from a smaller size nonlinear eigenvalue problem that is equivalent to the eigen-decomposition of $\mathbf{N}(\boldsymbol{\mu})$. The nonlinear eigenvalue problem is defined as

$$\mathbf{N}_s(\lambda_j(\boldsymbol{\mu}); \boldsymbol{\mu})\hat{\mathbf{q}}_{js}(\boldsymbol{\mu}) = \mathbf{0}, \tag{42}$$

where λ_j is an eigenvalue and $\hat{\mathbf{q}}_{js}$ the associated structural part of the eigenvector $\hat{\mathbf{q}}_j$. The matrix $\mathbf{N}_s \in \mathbb{R}^{2k_s \times 2k_s}$ is defined, in turn, as

$$\mathbf{N}_s(\lambda; \boldsymbol{\mu}) = \mathbf{N}_{ss}(\boldsymbol{\mu}) - \lambda\mathbf{I}_{2k_s} + \mathbf{N}_{sf}(\boldsymbol{\mu})\left(\lambda\mathbf{I}_{k_f} - \mathbf{N}_{ff}(\boldsymbol{\mu})\right)^{-1}\mathbf{N}_{fs}(\boldsymbol{\mu}), \tag{43}$$

where the blocks $\mathbf{N}_{ss} \in \mathbb{R}^{2k_s \times 2k_s}$, $\mathbf{N}_{sf} \in \mathbb{R}^{2k_s \times k_f}$, $\mathbf{N}_{ff} \in \mathbb{R}^{k_f \times k_f}$ and $\mathbf{N}_{fs} \in \mathbb{R}^{k_f \times 2k_s}$ are defined as

$$\mathbf{N}_{ss}(\boldsymbol{\mu}) = \begin{bmatrix} \mathbf{0} & -\boldsymbol{\Omega}_r^2(\boldsymbol{\mu}) \\ \mathbf{I}_{k_s} & \mathbf{0} \end{bmatrix}, \tag{44}$$

$$\mathbf{N}_{sf}(\boldsymbol{\mu}) = \begin{bmatrix} \mathbf{P}_r(\boldsymbol{\mu}) \\ \mathbf{0} \end{bmatrix}, \tag{45}$$

$$\mathbf{N}_{ff}(\boldsymbol{\mu}) = -\mathbf{H}_r(\boldsymbol{\mu}), \tag{46}$$

$$\mathbf{N}_{fs}(\boldsymbol{\mu}) = \begin{bmatrix} -\mathbf{R}_r(\boldsymbol{\mu}) & -\mathbf{G}_r(\boldsymbol{\mu}) \end{bmatrix}. \tag{47}$$

The nonlinear eigenvalue problem (42) can be solved either by a fixed-point iterative method or a continuation method as described in [6]. Once a structural eigenvalue λ_j is obtained, the corresponding damping ratio ζ_{js} is defined as

$$\zeta_{js} = -\frac{\lambda_j^R}{\sqrt{\left(\lambda_j^R\right)^2 + \left(\lambda_j^I\right)^2}}, \tag{48}$$

where λ_j^R and λ_j^I are real and imaginary part of λ_j, respectively.

In the context of solving the optimization problem (31), the operator $\mathbf{N}_s(\lambda; \boldsymbol{\mu})$ needs to be constructed and evaluated for various values of $\boldsymbol{\mu}$. However, $\mathbf{N}_s(\lambda_{S_l}; \boldsymbol{\mu})$ is not available for $\boldsymbol{\mu} \notin \mathcal{D}$. Thus the consistent ROM interpolation approach described in Section 5 is applied to construct $\mathbf{N}_s(\lambda; \boldsymbol{\mu})$ for $\boldsymbol{\mu} \notin \mathcal{D}$. Let $\mathbf{N}_f = \mathbf{N}_{sf}(\boldsymbol{\mu})\left(\lambda\mathbf{I}_{k_f} - \mathbf{N}_{ff}(\boldsymbol{\mu})\right)^{-1}\mathbf{N}_{fs}(\boldsymbol{\mu})$ and note that $\mathbf{N}_s = \mathbf{N}_{ss} - \lambda\mathbf{I}_{2k_s} + \mathbf{N}_f$ by (43). Note that the block $\mathbf{N}_{ss}$ is symmetric positive definite, so it is interpolated on the manifold of SPD matrices. The matrix $\mathbf{N}_f$ is a singular matrix. Thus $\mathbf{N}_f$ is interpolated on the manifold of square matrices $\mathbb{R}^{2k_s \times 2k_s}$.

In gradient-based optimization algorithms, the sensitivity of the damping ratio with respect to optimization parameter $\frac{d\zeta_s}{d\boldsymbol{\mu}}$ needs to be provided. The detailed derivation of $\frac{d\zeta_s}{d\boldsymbol{\mu}}$, using consistent interpolation on matrix manifolds is described in Appendix 8.

Table 3: ARW-2 specifications

Parameter	Value
Geometry (inch)	
Wingspan	104.9
Root	40.2
Tip	12.5
Material properties	
Skin (except flaps)	Various composite materials
Stiffeners	Aluminum
E (psi)	1.03×10^7
ρ (lbf·s^2/in^4)	2.6×10^{-4}
ν	0.32

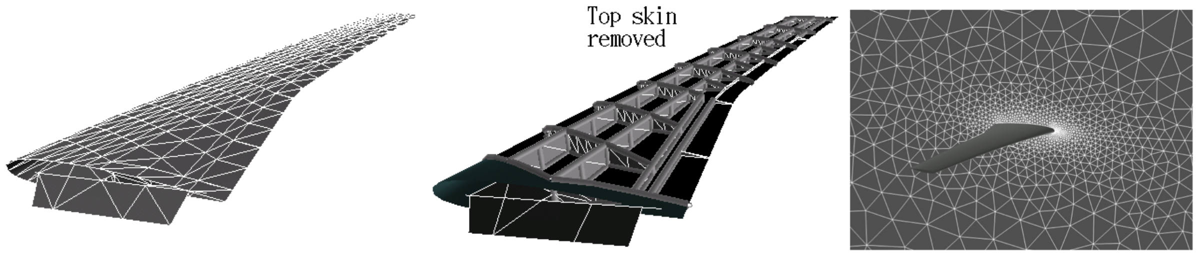


Figure 6: Finite element model for ARW-2 structure (left and center) and Fluid mesh around ARW-2 (right)

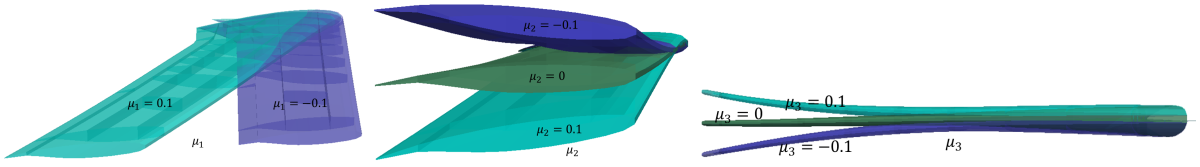


Figure 7: Variations of the shape variables

Table 4: Performance comparison of the various greedy algorithms considered.

Algorithm	*Classical*	*Random*	*Fixed* τ_s	*Adaptive* τ_s	*Surrogate*
N_Ξ	125	125	125	125	125
N_Π	-	20	20	20	10
Number of Sampled HDMs	16	15	15	13	15
Max. Rel. OIBEE (%)	4.9	3.9	4.6	4.7	4.6
Total Elapsed Time (hour)	60.9	12.1	8.8	8.4	7.3
Speed-up wrt Classical Greedy	1	5.1	6.9	7.3	8.3

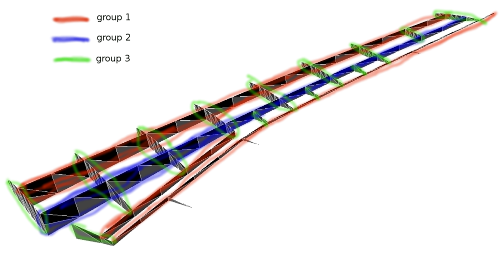


Figure 8: Three groups of stiffeners of ARW-2

6.2 Design optimization of the ARW2

The optimization problem (31) is solved for the Aeroelastic Research Wing (ARW-2) [33]. Physical dimension and material properties of the wing are reported in Table 3. A detailed FE model of the structure (Figure 6) is considered that includes, among others, spars, ribs, hinges, and control surfaces of the wing, and that contains a total of $2,556$ degrees of freedom. A three dimensional unstructured fluid mesh (Figure 6) around the wet surface with $63,484$ grid points is generated. An operating flight configuration is set for the altitude $h = 4,000$ ft with atmospheric density $\rho_\infty = 1.0193 \times 10^{-7}$ lb $\cdot$ s^2/in^4 and atmospheric pressure $P_\infty = 12.7$ lb/in^2, Mach number $M_\infty = 0.8$, a zero angle of attack.

The upper bound for the von Mises stress constraint (σ_{upper}) is 2.5×10^4 psi and the upper bound for the weight (W_{upper}) 400 lbs. The lower bound for the damping ratio (ζ_{lower}) is chosen between 4.1×10^{-4} and 4.5×10^{-4}. Three external wing shape parameters $\boldsymbol{\mu}_s \in \mathbb{R}^3$ and three structural parameters $\boldsymbol{\mu}_m \in \mathbb{R}^3$ are considered. The structural shape parameters include back sweep angle (μ_s^1), twist angle (μ_s^2), and dihedral angle (μ_s^3), which are depicted in Figure 7. The structural parameters ($\boldsymbol{\mu}_m$) are thickness increments for three disjoint groups of stiffeners. The groups of stiffeners are depicted in Figure 8.

Two distinct optimization problems are considered: 1) only the external shape parameters are used as optimization variables (i.e., $\boldsymbol{\mu} = \boldsymbol{\mu}_s$) 2) both external shape and structural parameters are considered (i.e., $\boldsymbol{\mu} = (\boldsymbol{\mu}_s, \boldsymbol{\mu}_m)$). Both cases are solved with the MATLAB fmincon implementation of the active set method. Both $\boldsymbol{\mu}_s$ and $\boldsymbol{\mu}_m$ are also normalized so that their upper and lower bounds are 0.1 and -0.1, respectively. For each problem, the proposed ROM database approach for handling the flutter constraints is compared to an optimization based on the HDM.

6.3 Offline database construction

Table 4 compares the performance of several greedy algorithms in the offline phase of the ROM-constrained optimization when $\boldsymbol{\mu} = \boldsymbol{\mu}_s$ (see Section 3). The candidate set is constructed by full factorial design with five points each axis, leading to $N_\Xi = 125$. The subset size N_Π is 20 and the size of the set for the sanity check 50. The marginal factor γ is 1 and $\epsilon_{tol} = 0.05$.

- *Classical* is for the classical greedy algorithm, in which error estimates for every candidate point is evaluated.
- *Random* denotes a greedy algorithm that chooses a random subset of the candidate set as described in Section 4.2.2.
- *Fixed* τ_s denotes a saturation-assumption based adaptive greedy algorithm where the fixed saturation constant $\tau_s = 2$ is used.
- *Adaptive* τ_s denotes a saturation-assumption based adaptive greedy algorithm where the saturation constant τ_s is modified at each greedy iteration according to Algorithm 1.
- *Surrogate* denotes an adaptive greedy algorithm where a surrogate surface of error estimate is used to pick a next point and add to the database (e.g., see [31]).

All the greedy algorithms use the One-Iteration-Based Error Estimate (OIBEE) as an error indicator proposed in [12]. There is a particular characteristic of evaluations of OIBEE for ARW-2 damping ratio when shape parameter is present. It is less expensive to evaluate OIBEE if a set of shape parameters is revisited because all the computations required to update the shape of ARW-2 have already been processed. *Classical*, *Random*, *Fixed*, and *Adaptive* greedy algorithms have higher probabilities of revisiting a point in parameter space than *Surrogate* because they work on the whole or a random subset of a fixed set of candidate points at each greedy iteration but *Surrogate* do not. This is why *Surrogate* is more expensive than other three adaptive greedy algorithms in Table 4. All the greedy algorithms except *Classical* have a randomness in choosing a subset of candidate set. The results shown in Table 4 are from one representative instance of greedy simulations. By taking a random subset of the candidate set, the speed-up of at least 5 is achieved from *Classical*. Further speed-up is achieved by using the saturation assumption filtering. A similar performance is achieved by *Fixed* and *Adaptive* τ_s (e.g., a speed-up of 6.9 for *Fixed* τ_s and 7.3 for *Adaptive* τ_s).

6.4 Speed-up for one online evaluation of the flutter constraint due to ROM database

Table 5 compares the performance of the HDM and the ROM database models for one online evaluation of the flutter constraint ($\boldsymbol{\zeta}$ and $\frac{d\boldsymbol{\zeta}}{d\boldsymbol{\mu}}$). Both wall-clock and CPU times are reported where CPU times are calculated as products of wall-clock time and the number of CPUs. For $\boldsymbol{\mu} = \boldsymbol{\mu}_s$, wall time speed-up of 16.7 and CPU time speed-up of 535.1 are achieved. For $\boldsymbol{\mu} = (\boldsymbol{\mu}_s, \boldsymbol{\mu}_m)$, wall time speed-up of 28.9 and CPU time speed-up of 926.1 are achieved. Both cases emphasize the large computational speed-up achieved by the proposed ROM database strategy.

6.5 Online predictions and optimization for $N_\mu = 3$

Table 6 presents the optimization results for $\boldsymbol{\mu} = \boldsymbol{\mu}_s$ and $\zeta_{\text{low}} = 4.2 \times 10^{-4}$. Both the HDM and the ROM database approach start from the same initial shape $\boldsymbol{\mu} = (0.1, 0.1, 0.1)$ and converge to similar optimized designs. The initial and optimized shapes of ARW-2 are depicted in Figure 9 for the HDM-based optimization. Although the optimized shape of ARW-2 for the ROM database approach-based optimization is not depicted in Figure 9, it is almost the same as the one for the HDM-based optimization. The flutter constraint is violated at the initial shape, but is satisfied at the optimal shape in both cases. The lift-drag ratio increases by 15.1%, the weight and the maximum von Mises stresses are raised by 4.4% and 18.2%. Although a similar number of optimization iterations is taken both for the HDM and the ROM database approach (5 and 6, respectively), the optimization path is different. Figures 10 and 11 show the ARW-2 shape changes corresponding to both optimization paths. Figure 12 shows the optimization iteration history of shape variable values and Figure 13 shows the optimization iteration history of various quantities such as lift-drag ratio, minimum damping ratio, maximum von Mises stress, and weight.

Table 6 also reports the CPU times and corresponding speed-ups. The Offline Phase reports the CPU hours required to construct a database of ROMs by a greedy algorithm using the adaptive τ_s

Table 5: Computational time (hour) for one online evaluation of the flutter constraint and its sensitivities

	$\boldsymbol{\mu} = \boldsymbol{\mu}_s$ $(N_\mu = 3)$			$\boldsymbol{\mu} = (\boldsymbol{\mu}_s, \boldsymbol{\mu}_m)$ $(N_\mu = 6)$		
	# of CPUs	Wall Time	CPU time	# of CPUs	Wall Time	CPU time
HDM	32	0.342	10.934	32	0.492	15.744
ROM database	1	0.019	0.019	1	0.017	0.017
Speed-up		18.0	575.5		28.9	926.1

approach and the Online Static Constraint Evaluation shows the CPU hours for computing all the static constraints and objective functions. The Online Flutter Constraint Evaluation shows the CPU hours of computing flutter constraints (i.e., damping ratio and its sensitivities). A speed-up of 559.3 is gained when comparing the Online Flutter Constraint Evaluation. A speed-up of 17.6 is achieved when the total online computation is compared. This smaller speed-up is due to the fact that the static constraint HDM is not reduced. In the present paper, the overall total computational time for the ROM-interpolation approach is larger than the one for the HDM due to the expensive cost of constructing a database ROM in the offline phase.

Table 6: CPU timings for an ARW-2 shape optimization problem with $\zeta_{\text{low}} = 4.2 \times 10^{-4}$ and $\boldsymbol{\mu} = \boldsymbol{\mu}_s$

Design	Initial	Optimized	Optimized
Approach		HDM	ROM Database
$\boldsymbol{\mu}_s$	(0.1,0.1,0.1)	(-0.1,-0.0814,0.1)	(-0.1,-0.0807,0.1)
L/D	10.6	12.2	12.2
Weight (lbs)	342.6	357.7	357.7
Min. $\boldsymbol{\zeta}$	2.24×10^{-4}	4.22×10^{-4}	4.25×10^{-4}
Max. σ_{VM} (psi)	18,731.9	22,139.5	22,139.0
Optimization Iterations		5	6
Function Evaluations		24	22
CPU time (hour)			
Offline Phase		0	268.8
Online Static Constraint Evaluation		14.2	14.1
Online Flutter Constraint Evaluation		240.5	0.43
Online Phase Total		254.7	14.5
Total (Offline+Online)		254.7	283.3
Speed-up			
Flutter Constraint Speed-up		1	559.3
Online Phase Speed-up		1	17.6
Total Speed-up		1	0.7

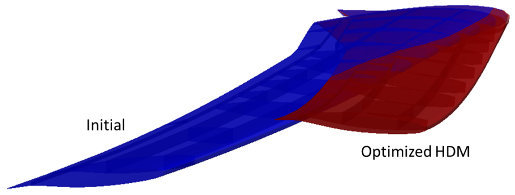


Figure 9: Initial and optimized configuration of ARW-2

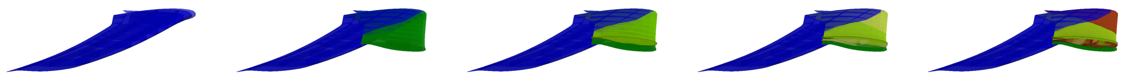


Figure 10: ARW-2 shape history of HDM optimization (Iterations 1 to 5 from the left to right)

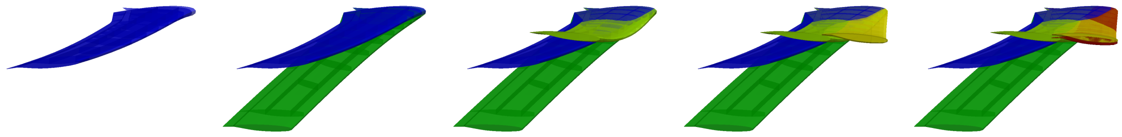


Figure 11: ARW-2 shape history of ROM optimization (Iterations 1, 2, 3, 4, 6 from the left to right)

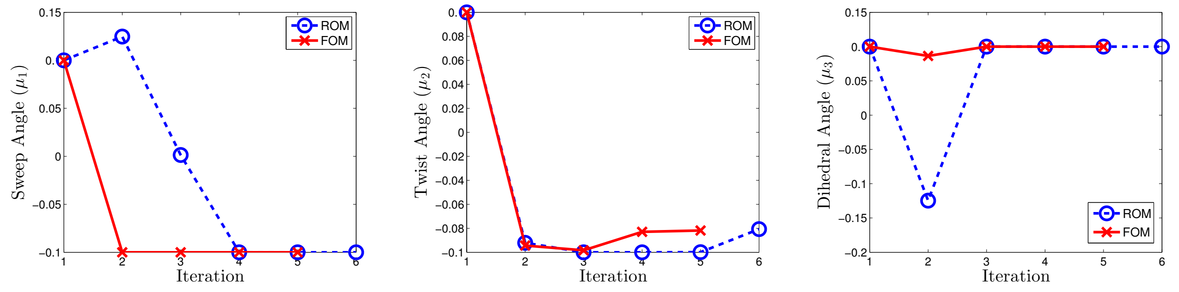


Figure 12: History of shape variables for the ARW-2 optimization

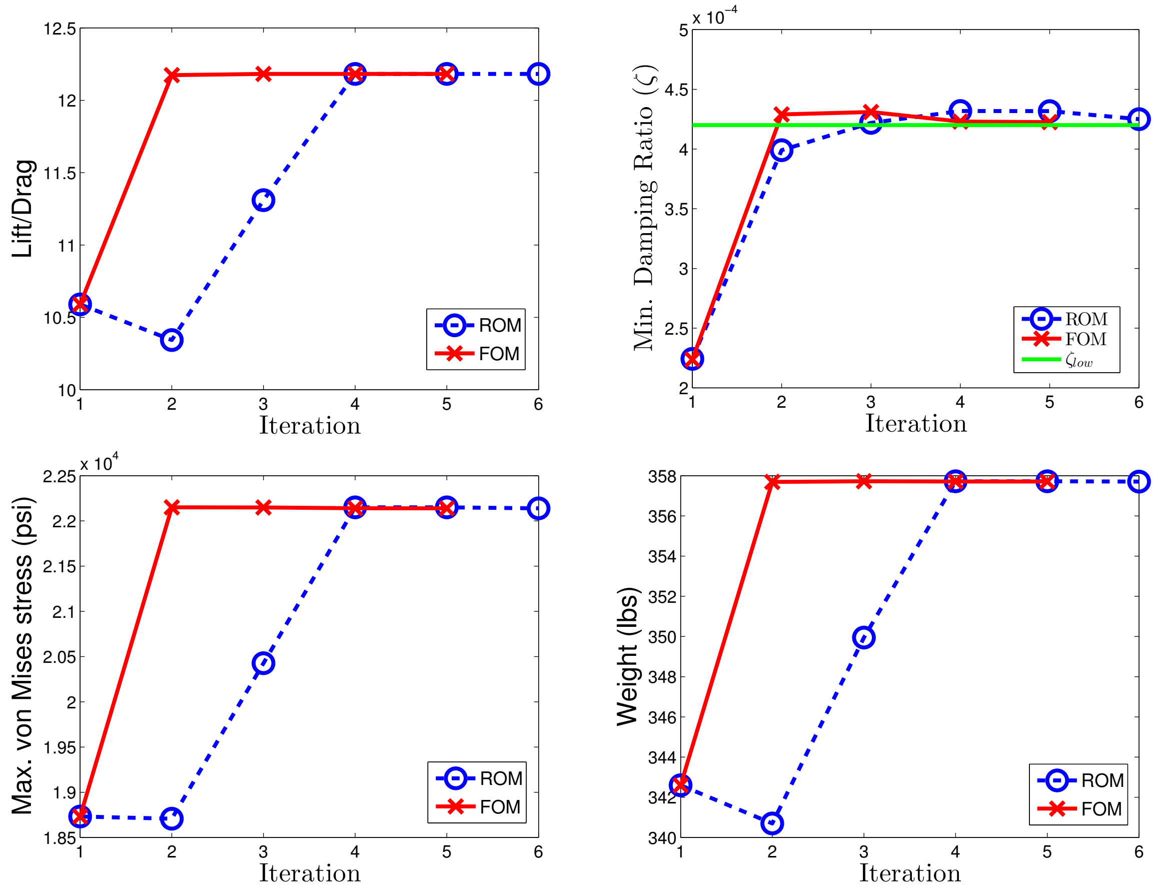


Figure 13: History of various quantities for the ARW-2 optimization

However, the database of ROMs can be reused for multiple optimization problems such as multi-start, multi-objective optimization, and robust optimization. In these contexts, the ROM-interpolation approach has the potential to lead to much more speed-up. Table 7 presents the CPU time for multiple flutter optimization problems of the form (31) with nine different values of lower bounds for the damping ratio ($\zeta_{\text{low}} = \{4.1, 4.15, 4.2, 4.25, 4.3, 4.35, 4.4, 4.45, 4.5\} \times 10^{-4}$). For each lower bound of damping ratio, a multi-start strategy with ten different initial points is applied to seek a global optimizer. Therefore, ninety independent optimization problems are solved using both the HDM and the ROM database approach. A speed-up of 1459.3 is achieved for the flutter constraint computations and the speed-up of 43.6 for the online phase. A speed-up of 17.1 is achieved even for the total CPU time including the offline phase of constructing a database ROM and online phase, Figure 14 shows the optimal lift-drag ratio found from the multiple flutter optimization solutions and the corresponding weights and the maximum von Mises stresses. The optimal lift-drag ratio decreases as the lower bound for the damping ratio increases. On the other hand, the corresponding maximum von Mises stress increases. Figure 15 (left) shows the corresponding minimum damping ratios. The ROM database is constructed by the saturation assumption-based greedy algorithm with the maximum OIBEE convergence threshold of 5%. Figure 15 (right) shows 5% error bar and actual relative errors of the ROM database model. The minimum damping ratio computed by the ROM database model falls much below the convergence threshold 5%.

Table 7: Total CPU time for multiple flutter optimization problems

	HDM	ROM Database
Speed-up (hours)		
Offline	0	268.8
Static Constraints	2,065.1	755.7
Dynamic Constraints	31,813.5	21.8
Miscellaneous Online	14.2	0.06
Online Total	33,892.8	777.6
Total	33,892.8	1,046.4
Speed-up		
Dynamic Speed-up	1	1,459.3
Online Speed-up	1	43.6
Total Speed-up	1	32.4

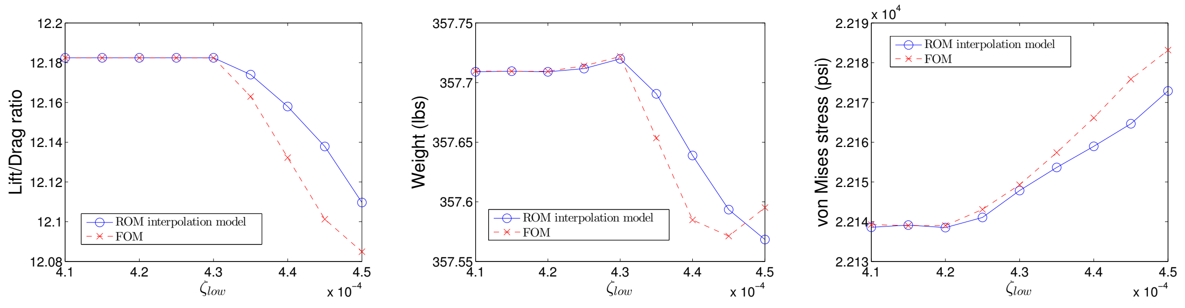
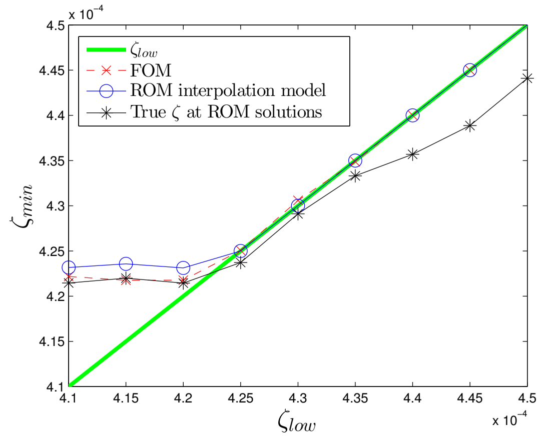


Figure 14: Optimal design dependence on the flutter constraint



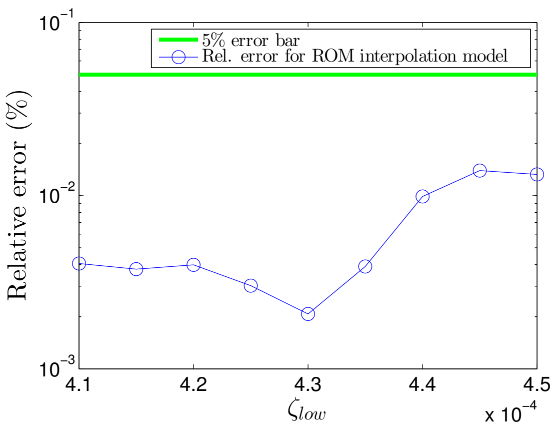


Figure 15: Minimum damping ratio of optimal design dependence on the flutter constraint

6.6 Online predictions and optimization for $N_\mu = 6$

Table 8 presents the optimization results for $\boldsymbol{\mu} = (\boldsymbol{\mu}_s, \boldsymbol{\mu}_m)$ and $\zeta_{\text{low}} = 4.2 \times 10^{-4}$. Both the HDM and the ROM database models start from the same initial point and lead to the same optimal shape $\boldsymbol{\mu}_s$. The two models, however, lead to slightly different optimal structure material parameters $\boldsymbol{\mu}_m$. The flutter constraint is violated at the initial design, but is satisfied at the optimized design. The lift-drag ratio increases by 8% and the weight is raised by 4.7% for the ROM database model and remains almost constant for the HDM. The maximum von Mises stress decreases by 6.6% for the ROM database model and 6.4% for the HDM. Although the optimal shape parameter $\boldsymbol{\mu}_s = (-0.1, -0.1, 0.1)$ is similar to the one in Table 6, the maximum von Mises stress for $\boldsymbol{\mu} = (\boldsymbol{\mu}_s, \boldsymbol{\mu}_m)$ is smaller than the maximum von Mises stress corresponding to the optimal solution for $\boldsymbol{\mu} = \boldsymbol{\mu}_s$. This is accomplished by increasing the thickness of the stiffeners. A similar number of optimization iterations and function evaluations are taken for the HDM and the ROM database model.

Table 8 also reports the CPU time and corresponding speed-ups. A speed-up of 1154.7 is gained if only the dynamic constraint computation is compared. A speed-up of 14.2 is achieved if the total online computation is compared. The total computational time for the ROM-interpolation approach is larger than the one for the HDM because of the expensive cost of constructing a database ROM in the offline phase. However, the ROM database model can be reused in multiple optimization solves such as robust optimization and multi-objectives optimization problems. Figure 16 shows the total CPU time speed-up dependence on the number of optimization solves under the assumption that one optimization solve for the ROM database model (or the HDM) takes the similar time to the one in Table 8. For example, 52 optimization solves give a total time speed-up of one. The total CPU time speed-up converges as the number of optimization solves increases to 14.2, which is the online time speed-up.

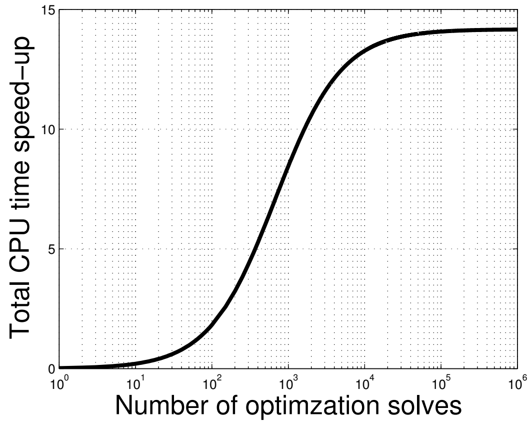


Figure 16: Total CPU time speed-up dependence on the number of optimization solves

7 Conclusions

A novel methodology for a solution of PDE-constrained optimization problems is introduced. A database of local ROMs is constructed and consistent ROM interpolation is performed to accelerate the optimization phase. The methodology is applied to the design optimization of a realistic aeroelastic wing. A large online CPU time speed-up is achieved. The online flexibility of the ROM database approach results in the applicability to multiple-objectives optimization, robust optimization, and multi-start strategies for a global optimization. In the context of multiple solutions of optimization, in turn, a total CPU time speed-up can also be gained. The accuracy of the ROM database model can be tuned in the offline phase where a database of local ROMs is constructed by a greedy procedure. A novel greedy algorithm based on the saturation assumption, in which a value of the saturation constant is adaptively updated in each

Table 8: CPU timings for the ARW-2 shape optimization problem with $\zeta_{\text{low}} = 4.2 \times 10^{-4}$ and $\boldsymbol{\mu} = (\boldsymbol{\mu}_s, \boldsymbol{\mu}_m)$

Design	Initial	Optimized	Optimized
Approach		HDM	ROM Database
$\boldsymbol{\mu}_s$	(0,0,0)	(-0.1,-0.1,0.1)	(-0.1,-0.1,0.1)
$\boldsymbol{\mu}_m$	(0,0,0)	(0.1,0.1,-0.015)	(0.1,0.1,0.013)
L/D	11.3	12.2	12.2
Weight (lbs)	349.9	349.8	366.3
Min. $\boldsymbol{\zeta}$	3.7×10^{-4}	4.2×10^{-4}	4.2×10^{-4}
Max. σ_{VM} (psi)	20,297.1	18,991.9	18,953.3
Optimization Iterations		6	5
Function Evaluations		11	9
CPU time (hour)			
Offline Phase		0	8,800
Online Static Constraint Evaluation		11.0	12.8
Online Flutter Constraint Evaluation		173.2	0.15
Online Phase Total		184.2	13.0
Total (Offline+Online)		184.2	8,813
Speed-up			
Flutter Constraint Speed-up		1	1154.7
Online Phase Speed-up		1	14.2
Total Speed-up		1	0.02

iteration, is also introduced and shown competitive performance compared to other greedy algorithms.

8 Appendix

Differentiating (42) with respect to $\mu_i,\ i = 1, \cdots, N_\mu$ leads to

$$\frac{\partial \mathbf{N}_s}{\partial \mu_i}\mathbf{q}_{js} + \frac{\partial \mathbf{N}_s}{\partial \lambda_j}\frac{\partial \lambda_j}{\partial \mu_i}\mathbf{q}_{js} + \mathbf{N}_s \frac{\partial \mathbf{q}_{js}}{\partial \mu_i} = \mathbf{0}. \tag{49}$$

Multiplying by the left eigenvector $\mathbf{p}_{js_l}^H$ from the left to (49) and noting that $\mathbf{p}_{js_l}^H \mathbf{N}_s = \mathbf{0}$ leads to

$$\frac{\partial \lambda_j}{\partial \mu_i} = -\frac{\mathbf{p}_{js_l}^H \dfrac{\partial \mathbf{N}_s}{\partial \mu_i}\mathbf{q}_{js}}{\mathbf{p}_{js_l}^H \dfrac{\partial \mathbf{N}_s}{\partial \lambda_j}\mathbf{q}_{js}}. \tag{50}$$

The sensitivity of the damping ratio ζ_{js} with respect to μ_i is then obtained by chain rule:

$$\frac{\partial \zeta_{js}}{\partial \mu_i} = \frac{\partial \lambda_j^R}{\partial \mu_i}\left(\frac{(\lambda_j^R)^2}{|\lambda_j|^3} - \frac{1}{|\lambda_j|}\right) + \frac{\partial \lambda_j^I}{\partial \mu_i}\frac{\lambda_j^R \lambda_j^I}{|\lambda_j|^3}. \tag{51}$$

Taking the derivative of (43) with respect to μ_i leads to

$$\frac{\partial \mathbf{N}_s}{\partial \mu_i}(\lambda_j, \boldsymbol{\mu}^\star) = \frac{\partial \mathbf{N}_f}{\partial \mu_i}(\lambda_j, \boldsymbol{\mu}^\star) + \frac{\partial \mathbf{N}_{ss}}{\partial \mu_i}(\boldsymbol{\mu}^\star). \tag{52}$$

The sensitivity $\dfrac{\partial \mathbf{N}_f}{\partial \mu_i}$ of the operator $\mathbf{N}_f$ interpolated on the matrix manifold $\mathbb{R}^{2k_s \times 2k_s}$ can be obtained by (25) and Tables 1 and 2:

$$\frac{\partial \mathbf{N}_f}{\partial \mu_i}(\lambda_j, \boldsymbol{\mu}^\star) = \frac{d\mathcal{I}}{d\mu_i}\left(\boldsymbol{\mu}^\star; \{\mathbf{N}_f(\lambda_j, \boldsymbol{\mu}_c) - \mathbf{N}_f(\lambda_j, \boldsymbol{\mu}_1)\}_{c=1}^{N_p}\right). \tag{53}$$

The sensitivity $\dfrac{\partial \mathbf{N}_{ss}}{\partial \mu_i}$ of the operator $\mathbf{N}_{ss}$ interpolated on the manifold of SPD matrices is

$$\frac{\partial \mathbf{N}_{ss}}{\partial \mu_i}(\boldsymbol{\mu}^\star) = \mathbf{N}_{ss_1}^{1/2} \frac{\partial \exp\left(\mathcal{I}\left(\boldsymbol{\mu}^\star; \{\log(\mathbf{N}_{ss_1}^{-1/2}\mathbf{N}_{ss_c}\mathbf{N}_{ss_1}^{-1/2}\}_{c=1}^{N_p}\right)\right)}{\partial \mu_i} \mathbf{N}_{ss_1}^{1/2}, \tag{54}$$

where the derivative of the exponential matrix can be obtained by (27). Taking the derivative of (43) with respect to λ_j leads to

$$\frac{\partial \mathbf{N}_s}{\partial \lambda_j}(\lambda_j, \boldsymbol{\mu}^\star) = \frac{\partial \mathbf{N}_f}{\partial \lambda_j}(\lambda_j, \boldsymbol{\mu}^\star) - \mathbf{I}. \tag{55}$$

Since $\mathbf{N}_f(\lambda_j, \boldsymbol{\mu}^\star) = \mathbf{N}_f(\lambda_j, \boldsymbol{\mu}_1) + \mathcal{I}\left(\boldsymbol{\mu}^\star; \{\mathbf{N}_f(\lambda_j, \boldsymbol{\mu}_c) - \mathbf{N}_f(\lambda_j, \boldsymbol{\mu}_1)\}_{c=1}^{N_p}\right)$,

$$\begin{aligned}\frac{\partial \mathbf{N}_f}{\partial \lambda_j}(\lambda_j, \boldsymbol{\mu}^\star) &= \frac{\partial \mathbf{N}_f}{\partial \lambda_j}(\lambda_j, \boldsymbol{\mu}_1) + \frac{\partial \mathcal{I}}{\partial \lambda_j}(\boldsymbol{\mu}^\star; \{\mathbf{N}_f(\lambda_j, \boldsymbol{\mu}_c) - \mathbf{N}_f(\lambda_j, \boldsymbol{\mu}_1)\}_{c=1}^{N_p}) \\ &= \frac{\partial \mathbf{N}_f}{\partial \lambda_j}(\lambda_j, \boldsymbol{\mu}_1) + \mathcal{I}\left(\boldsymbol{\mu}^\star; \left\{\frac{\partial \mathbf{N}_f(\lambda_j, \boldsymbol{\mu}_c)}{\partial \lambda_j} - \frac{\partial \mathbf{N}_f(\lambda_j, \boldsymbol{\mu}_1)}{\partial \lambda_j}\right\}_{c=1}^{N_p}\right).\end{aligned} \tag{56}$$

The second equality in (56) is true if the interpolation operator $\mathcal{I}\left(\boldsymbol{\mu}^\star; \{\Gamma_c\}_{c=1}^{N_p}\right)$ is linear in Γ_c, $c = 1, \ldots, N_p$. Indeed, this property is true for many interpolation operators, including polynomial and RBF (Radial Basis Function)-based interpolations. Note also that $\mathbf{N}_f(\lambda_j, \boldsymbol{\mu}_c) = \mathbf{N}_{sf}(\boldsymbol{\mu}_c)(\lambda_j \mathbf{I} - \mathbf{N}_{ff}(\boldsymbol{\mu}_c))^{-1}\mathbf{N}_{fs}(\boldsymbol{\mu}_c)$ for $c = 1, \cdots, N_p$. Hence

$$\frac{\partial \mathbf{N}_f}{\partial \lambda_j}(\lambda_j, \boldsymbol{\mu}_c) = -\mathbf{N}_{sf}(\boldsymbol{\mu}_c)(\lambda_j \mathbf{I} - \mathbf{N}_{ff}(\boldsymbol{\mu}_c))^{-2}\mathbf{N}_{fs}(\boldsymbol{\mu}_c). \tag{57}$$

The derivative $\frac{\partial \mathbf{N}_s}{\partial \lambda_j}$ is then obtained by plugging (57) in (55).

9 Acknowledgements

The authors acknowledge the support from the office of Naval Research under Grant N00014-11-1-0707. The content of this publication does not necessarily reflect the position or policy of any of these supporters, and no official endorsement should be inferred.

References

[1] David Amsallem, Julien Cortial, Kevin Carlberg, and Charbel Farhat. A method for interpolating on manifolds structural dynamics reduced-order models. *International journal for numerical methods in engineering*, 80(9):1241–1258, 2009.

[2] David Amsallem and Charbel Farhat. Interpolation method for adapting reduced-order models and application to aeroelasticity. *AIAA journal*, 46(7):1803–1813, 2008.

[3] David Amsallem and Charbel Farhat. An online method for interpolating linear parametric reduced-order models. *SIAM J. Sci. Comput.*, 33(5):2169–2198, 2011.

[4] David Amsallem and Charbel Farhat. On the stability of reduced-order linearized computational fluid dynamics models based on POD and Galerkin projection: descriptor vs non-descriptor forms. In *Reduced Order Methods for Modeling and Computational Reduction*, pages 215–233. Springer, 2014.

[5] David Amsallem and Ulrich Hetmaniuk. A posteriori error estimators for linear reduced-order models using krylov-based integrators. *International Journal for Numerical Methods in Engineering*, 102(5):1238–1261, May 2015.

[6] David Amsallem, Daniel Neumann, Youngsoo Choi, and Charbel Farhat. Linearized aeroelastic computation in the frequency domain based on computational fluid dynamics. *submitted to Arxiv*, pages 1–23, June 2015.

[7] David Amsallem, Radek Tezaur, and Charbel Farhat. Real-time solution of computational problems using databases of parametric linear reduced-order models with arbitrary underlying meshes. *submitted to Arxiv*, pages 1–30, June 2015.

[8] David Amsallem, Matthew Zahr, Youngsoo Choi, and Charbel Farhat. Design optimization using hyper-reduced-order models. *Structural and Multidisciplinary Optimization*, 51(4):919–940, 2015.

[9] Tan Bui-Thanh, Karen Willcox, and Omar Ghattas. Model reduction for large-scale systems with high-dimensional parametric input space. *SIAM Journal on Scientific Computing*, 30(6):3270–3288, 2008.

[10] Tan Bui-Thanh, Karen Willcox, and Omar Ghattas. Parametric reduced-order models for probabilistic analysis of unsteady aerodynamic applications. *AIAA journal*, 46(10):2520–2529, 2008.

[11] Youngsoo Choi. *Simultaneous Analysis and Design in PDE-constrained Optimization.* PhD thesis, Stanford University, 2012.

[12] Youngsoo Choi, David Amsallem, Arthur Paul-Dubois-Taine, and Charbel Farhat. Novel error estimates for constructing a database of reduced order models in aeroelastic nonlinear eigenvalue problems. submitted for publication, 2015.

[13] Youngsoo Choi, Charbel Farhat, Walter Murray, and Michael Saunders. A practical factorization of a schur complement for pde-constrained distributed optimal control. *Journal of Scientific Computing*, pages 1–22, 2014.

[14] Andrew R Conn, Nicholas IM Gould, and Ph L Toint. *Trust region methods*, volume 1. Siam, 2000.

[15] Bogdan Epureanu. A parametric analysis of reduced order models of viscous flows in turbomachinery. *Journal of fluids and structures*, 17(7):971–982, 2003.

[16] Marco Fahl and Ekkehard W Sachs. Reduced order modelling approaches to PDE-constrained optimization based on proper orthogonal decomposition. In *Large-scale PDE-constrained optimization*, pages 268–280. Springer, 2003.

[17] Charbel Farhat, Michel Lesoinne, and Nathan Maman. Mixed explicit/implicit time integration of coupled aeroelastic problems: Three-field formulation, geometric conservation and distributed solution. *International Journal for Numerical Methods in Fluids*, 21:807–835, 1995.

[18] Roland W Freund. Model reduction methods based on krylov subspaces. *Acta Numerica*, 12:267–319, 2003.

[19] Philip E Gill, Walter Murray, and Michael A Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12:979–1006, 2002.

[20] Gene H. Golub and Charles F. Van Loan. *Matrix Computations. 1996.* Johns Hopkins University Press, 1996.

[21] Martin A Grepl and Anthony T Patera. A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 39(01):157–181, 2005.

[22] Bernard Haasdonk and Mario Ohlberger. Reduced basis method for finite volume approximations of parametrized linear evolution equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 42(02):277–302, 2008.

[23] Bernard Haasdonk and Mario Ohlberger. Efficient reduced models and a posteriori error estimation for parametrized dynamical systems by offline/online decomposition. *Mathematical and Computer Modelling of Dynamical Systems*, 17(2):145–161, 2011.

[24] Jan S Hesthaven, Benjamin Stamm, and Shun Zhang. Efficient greedy algorithms for high-dimensional parameter spaces with applications to empirical interpolation and reduced basis methods. *ESAIM: Mathematical Modelling and Numerical Analysis*, 48(01):259–283, 2014.

[25] Taehyoun Kim. Frequency-domain Karhunen-Loeve method and its application to linear dynamic systems. *AIAA Journal*, 36(11):2117–2123, 1998.

[26] Patrick A LeGresley and Juan Alonso. Airfoil design optimization using reduced order models based on proper orthogonal decomposition. In *AIAA Paper 2000-2545 Fluids 2000 Conference and Exhibit, Denver, CO*, pages 1–14, 2000.

[27] Michel Lesoinne, Marcus Sarkis, Ulrich Hetmaniuk, and Charbel Farhat. A linearized method for the frequency analysis of three-dimensional fluid/structure interaction problems in all flow regimes. *Computer Methods in Applied Mechanics and Engineering*, 190:3121–3146, 2001.

[28] Kurt Maute, Melike Nikbay, and Charbel Farhat. Coupled analytical sensitivity analysis and optimization of three-dimensional nonlinear aeroelastic systems. *AIAA journal*, 39(11):2051–2061, 2001.

[29] Bruce Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *Automatic Control, IEEE Transactions on*, 26(1):17–32, 1981.

[30] Igor Najfeld and Timothy F. Havel. Derivatives of the matrix exponential and their computation. *Advances in Applied Mathematics*, 16(3):321–375, 1995.

[31] Arthur Paul-Dubois-Taine and David Amsallem. An adaptive and efficient greedy procedure for the optimal training of parametric reduced-order models. *International Journal for Numerical Methods in Engineering*, 102(5):1262–1292, May 2015.

[32] Christophe Prudhomme, Dimitrios V Rovas, Karen Veroy, Luc Machiels, Yvon Maday, Anthony T Patera, and Gabriel Turinici. Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods. *Journal of Fluids Engineering*, 124(1):70–80, 2002.

[33] Maynard C Sandford, David A Seidel, Clinton V Eckstrom, and Charles V Spain. Geometrical and structural properties of an aeroelastic research wing (arw-2). 1989.

[34] Lawrence Sirovich. Turbulence and the dynamics of coherent structures. part i: Coherent structures. *Quarterly of applied mathematics*, 45(3):561–571, 1987.

[35] Karen Veroy, Christophe Prudhomme, Dimitrios V Rovas, and Anthony T Patera. A posteriori error bounds for reduced-basis approximation of parametrized noncoercive and nonlinear elliptic partial differential equations. In *Proceedings of the 16th AIAA computational fluid dynamics conference*, volume 3847, pages 23–26, 2003.

[36] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[37] Gary Weickum, Mike Eldred, and Kurt Maute. A multi-point reduced-order modeling approach of transient structural dynamics with application to robust design optimization. *Structural and Multidisciplinary Optimization*, 38(6):599–611, 2009.

[38] Yao Yue and Karl Meerbergen. Accelerating optimization of parametric linear systems by model order reduction. *SIAM Journal on Optimization*, 23(2):1344–1370, 2013.

[39] Matthew J Zahr, David Amsallem, and Charbel Farhat. Construction of parametrically-robust cfd-based reduced-order models for pde-constrained optimization. In *AIAA Paper 2013-2845, 21st AIAA Computational Fluid Dynamics Conference, San Diego, CA, June 26-29, 2013*, pages 1–11, Reston, Virginia, June 2013. American Institute of Aeronautics and Astronautics.

[40] Matthew J Zahr and Charbel Farhat. Progressive construction of a parametric reduced-order model for PDE-constrained optimization. *International Journal for Numerical Methods in Engineering*, 102(5):1111–1135, June 2015.